ANL-77-49

ANL-77-49

UPDATING THE SYMMETRIC INDEFINITE FACTORIZATION WITH APPLICATIONS IN A MODIFIED NEWTON'S METHOD

by

Danny C. Sorensen

U of C-AUA-USERDA

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Prepared for the U. S. ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION under Contract W-31-109-Eng-38

Printed in the United States of America
Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22161
Price: Printed Copy $6.75; Microfiche $3.00

Distribution Category:
Mathematics and Computers
(UC-32)

ANL-77-49

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

UPDATING THE SYMMETRIC INDEFINITE FACTORIZATION WITH APPLICATIONS IN A MODIFIED NEWTON'S METHOD

by

Danny C. Sorensen

Applied Mathematics Division

Based on a thesis prepared for
the degree Doctor of Philosophy in Mathematics
from the University of California, San Diego

June 1977

TABLE OF CONTENTS

Page

List of Figures 4
List of Tables. 4
List of Symbols 5
Abstract. 7

I. An Overview 9

1. Introduction. 9
2. Newton-type Methods for Unconstrained Optimization. 10
3. The Symmetric Indefinite Decomposition. 17
4. Computational Results and Conclusions 19

II. Updating Factorizations of Symmetric Matrices 21

1. Introduction. 21
2. Description of the Algorithm. 24
3. A Pictorial Description of the Algorithm. 49

III. Error Analysis of the Updating Algorithm. 55

1. Introduction. 55
2. A Detailed Description of the Updating Algorithm. 57
3. Floating Point Analysis 65

IV. The Use of Directions of Negative Curvature in a Modified Newton Iteration. 95

1. Introduction. 95
2. Descent Directions. 97
3. A Modification of the Armijo Steplength Procedure 100
4. Determining Directions of Negative Curvature. 103
5. A Steplength Algorithm. 107
6. Convergence of the Modified Newton Iteration. 113
7. Conclusions 119

V. Computational Results 121

1. Introduction. 121
2. Testing the Updating Algorithm. 121
3. Testing the Modified Newton's Method. 126

Appendix A1 139
Appendix A2 141
Acknowledgments 153
References. 154

LIST OF FIGURES

1 Pivoting in the Updating Algorithm 49

2 A Search Along $x+\alpha s$. 109

3 A Search Along $x+\alpha^2 s+\alpha d$. 110

4 The Curve $x+\alpha^2 s+\alpha d$ 111

LIST OF TABLES

1 Operations Required at Step k. 46

2 Results for Increasing Order 124

3 Results of a Long Range of Updates 125

4 Results of Tests with Standard Starts. 132

5 Results from Using Finite Differences. 133

6 Box's Function 134

7 EXP6 135

8 Gottfried's Function 136

9 Brown's Badly Scaled Problem 137

List of Symbols

R	The set of real numbers.
R^n	The n-dimensional real vector space.
$R^{m\times n}$	The space of m×n real matrices.
$\{\ \}$	The set of.
$\in$	Element inclusion.
$\subset$	Set inclusion.
x^t, A^t	The transpose of a vector x or matrix A.
$x^t y$	The inner product of two vectors $x,y \in R^n$.
$\|\cdot\|$	The Euclidean norm; $\|x\| = \sqrt{x^t x}$
$\|\cdot\|_\infty$	The infinity norm; $\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} \|a_{ij}\|$.
$\|\cdot\|$	The absolute value function.
sgn(•)	The algebraic sign function.
$f: \mathcal{D} \to X$	A mapping f from a domain $\mathcal{D}$ into the set X.
$\partial_k f(x)$	The first partial derivative of f at x with respect to the k-th argument.
$\partial_{ij} f(x)$	The second partial derivative of f at $x \in R^n$ with respect to the i-th and then the j-th argument.
A => B	A implies B.
$a_k \to b$	a_k approaches b.

UPDATING THE SYMMETRIC INDEFINITE FACTORIZATION WITH APPLICATIONS IN A MODIFIED NEWTON'S METHOD

by

Danny C. Sorensen

ABSTRACT

In recent years the use of quasi-Newton methods in optimization algorithms has inspired much of the research in an area of numerical linear algebra called updating matrix factorizations. Previous research in this area has been concerned with updating the factorization of a symmetric positive definite matrix. Here, a numerical algorithm is presented for updating the Symmetric Indefinite Factorization of Bunch and Parlett. The algorithm requires only $O(n^2)$ arithmetic operations to update the factorization of an $n \times n$ symmetric matrix when modified by a rank one matrix. An error analysis of this algorithm is given. Computational results are presented that investigate the timing and accuracy of this algorithm.

Another algorithm is presented for the unconstrained minimization of a nonlinear functional. The algorithm is a modification of Newton's method. At points where the Hessian is indefinite the search for the next iterate is conducted along a quadratic curve in the plane spanned by a direction of negative curvature and a gradient related descent direction. The stopping criteria for this search take into account the second order derivative information. The result is that the iterates are shown to converge globally to a critical point at which the Hessian is positive semidefinite. Computational results are presented which indicate that the method is promising.

Chapter I

An Overview

1. Introduction

In recent years the use of matrix methods in optimization algorithms has received an increasing amount of attention. Interesting problems in numerical linear algebra have been generated by advances in optimization methods. Similarly, new approaches to optimization methods are sometimes made possible or even suggested by advances in numerical linear algebra. Here the Bunch-Parlett factorization of a symmetric indefinite matrix is used in a Newton-type method which is based on the use of directions of negative curvature. In anticipation of the extension of these ideas for use in a quasi-Newton method, we present and analyze a method for updating this matrix factorization.

In this chapter the problems which shall be considered are introduced and motivated. Chapters II and III are concerned with the updating algorithm and should be considered as a unit. On the other hand, Chapter IV is meant to be self-contained. For this reason some of the same concepts are introduced in both places. The numbering of equations is done separately in each chapter. For example, a reference within a chapter to equation (2.1) means to refer to the equation numbered (2.1) which will be found in Section 2 of that chapter. Whenever there is a cross reference between chapters it will be explicitly mentioned.

2. Newton-type Methods for Unconstrained Optimization

One of the major problem areas of numerical analysis is the minimization of a non-linear functional. If we denote the n-dimensional real vector space by R^n and the real numbers by R, the problem is:

given a domain $\mathcal{D} \subset R^n$ and a functional

$$f: \mathcal{D} \to R$$

find $x^* \in \mathcal{D}$ such that

$$f(x^*) \leq f(x)$$

for all $x \in \mathcal{D}$.

Usually the task of trying to find a global minimum of f is too difficult numerically, and we must be content with finding a local minimum for f. That is, we seek

$$x^* \in \mathcal{D}$$

such that

$$f(x) \leq f(x^*)$$

for all $x \in N(x^*) \subset D$ where $N(x^*)$ is some neighborhood of x^*.

Let

$$g(x) \equiv \begin{pmatrix} \partial_1 f(x) \\ \partial_2 f(x) \\ \vdots \\ \partial_n f(x) \end{pmatrix} = \nabla f(x)$$

be the gradient of f at x, and let

$$G(x) \equiv \begin{pmatrix} \partial_{11}f(x) & \partial_{12}f(x) \ldots . \partial_{1n}f(x) \\ \partial_{21}f(x) & \partial_{22}f(x) \ldots . \partial_{2n}f(x) \\ \vdots & \vdots & \vdots \\ \partial_{n1}f(x) & \partial_{n2}f(x) \ldots . \partial_{nn}f(x) \end{pmatrix}$$

$$= \nabla^2 f(x)$$

be the Hessian of f at x. For a sequence $\{x_k\}$ we shall write

$$f_k = f(x_k),$$

$$g_k = g(x_k),$$

$$G_k = G(x_k).$$

Assume that f has two continuous derivatives on D. Then the Hessian matrix is symmetric, and for any x, $\bar{x} \in D$ we have

$$f(\bar{x}) = f(x) + g(x)^t(\bar{x}-x) + \frac{1}{2}(\bar{x}-x)^t G(x)(\bar{x}-x) + o(\|\bar{x}-x\|^2).$$

We write $h(\varepsilon) = o(\varepsilon)$ if $\lim_{\varepsilon \to 0} \frac{h(\varepsilon)}{\varepsilon} = 0$. Thus f is modeled well locally by the quadratic form defined by the first three terms of its Taylor expansion about x. If the Hessian G(x) is positive definite then the quadratic form

$$f(x) + g(x)^t(\bar{x}-x) + \frac{1}{2}(\bar{x}-x)^t G(x)(\bar{x}-x)$$

has a minimum at

$$\bar{x} = x - G^{-1}(x)g(x). \tag{2.1}$$

Formula (2.1) suggests the iteration

(2.2) Given $x_0 \in \mathcal{D}$

for k=0,1,2,...

$$G_k s_k = -g_k$$

$$x_{k+1} = x_k + s_k$$

This is, of course, the well known Newton's method for finding a zero of the gradient $g(x)$. Thus Newton's method can be viewed as minimizing the local quadratic model of f and also as attempting to find a point x^* which satisfies $g(x^*) = 0$. This is important since

(2.3) f has a local minimum at x^* only if $g(x^*) = 0$.

This method has two important properties that make it a very powerful tool for the solution of unconstrained minimization problems. The first of these is the basic simplicity of the iteration (2.2). The second and most important property of Newton's method is the local quadratic rate of convergence of the iterates. Loosely stated this means that when the iterates x_k of (2.2) converge to a point x^* with $G(x^*)$ nonsingular, then eventually the number of significant digits in the approximant x_k doubles at each iteration. The more precise mathematical statement is contained in the following theorem. Before the theorem is stated it will be necessary to introduce the notion of a point of attraction. A point x^* is a point of attraction for the iteration (2.2) if there is an open neighborhood $N(x^*) \subset \mathcal{D}$ such that when $x_0 \in N(x^*)$, the iterates defined by (2.2) all lie in $\mathcal{D}$ and converge to x^*.

Theorem (2.1)

Assume that $g: \mathcal{D} \subset R^n \to R^n$ is continuously differentiable on an open neighborhood $N(x^*) \subset \mathcal{D}$ of a point $x^* \in \mathcal{D}$ for which $g(x^*) = 0$, and $G(x^*)$ is nonsingular. Then x^* is a point of attraction of the iteration (2.2). If, in addition, there exists a positive constant L such that $\|G(x) - G(x^*)\| \leq L\|x-x^*\|$ for all $x \in N(x^*)$, then there exists a positive constant C and a positive integer K such that $k > K$ implies that

$$\|x_{k+1} - x^*\| \leq C\|x_k - x^*\|^2 .$$

A proof of Theorem (2.1) can be found in [17, p. 312].

There are some major difficulties in implementing Newton's method in its basic form. The first of these difficulties is that there is no reason for the Hessian to be positive definite at an iterate x_k which is far from a local minimum. Another difficulty is that the step s_k predicted by the quadratic model at x_k may be too large or too small.

These difficulties have led to several modifications of Newton's method. Many of the modifications have taken the form

(2.4)
Given $x_0 \in \mathcal{D}$
for k=0,1,2,...
$\hat{G}_k = G_k + E_k$
$\hat{G}_k s_k = -g_k$
$x_{k+1} = x_k + \alpha_k s_k$.

The symmetric matrix E_k in (2.4) is chosen to insure that $\hat{G}_k$ is positive definite. This implies that the direction s_k satisfies

(2.5) $$g_k^t s_k < 0 .$$

Thus the directional derivative of f at x_k in the direction s_k is negative and the function must decrease initially in the direction s_k. A direction s_k that satisfies (2.5) is called a descent direction. Once a descent direction s_k has been specified it is possible to determine a positive number α_k such that $f(x_k+\alpha_k s_k) < f_k$.

Of course, the particular way in which the matrix E_k and the

scalar α_k are determined are crucial in analyzing the convergence of the iteration (2.4). Some success has been achieved with iterations of type (2.4) in specifying E_k, α_k in such a way that the iterates x_k are globally convergent to a critical point x^* (i.e. a point x^* with $g(x^*) = 0$). Whenever possible these algorithms reduce naturally to Newton's method so that the local quadratic rate of convergence is retained.

However, work in this area is not yet complete. In particular, no algorithm has yet been given which can guarantee global convergence to a local minimum. How can Newton's method be modified so that the resulting iterates converge globally to a local minimum for f? In attempting to answer this question, we have developed an algorithm which is different from iterations of type (2.4). This algorithm is presented and analyzed in Chapter IV. The algorithm is based more explicitly on the local quadratic model for f in that the Hessian is not modified. Instead, directions of negative curvature are used in combination with the more usual descent directions. The resulting iterates $\{x_k\}$ are shown to be globally convergent to a point x^* such that $g(x^*) = 0$, and $G(x^*)$ is positive semi-definite. Thus by basing the iteration more closely on the quadratic model we obtain an iteration which converges to a point x^* that satisfies the second order necessary conditions

(2.6) f has a local minimum at x^* only if $g(x^*) = 0$, and $G(x^*)$ is positive semi-definite.

Yet another drawback to a modified Newton's method is the expense in terms of both computation and programming effort associated with calculating the Hessian at each step of the iteration (2.4). Attempts to overcome this undesirable feature have led to a great deal

of research in a class of methods called quasi-Newton methods. These methods replace the Hessian G_k with an approximation B_k. A quasi-Newton iteration has the form

(2.7) Given $x_0 \in \mathcal{D}$, and B_0

for k=0,1,2,...

$$B_k s_k = -g_k$$
$$x_{k+1} = x_k + \alpha_k s_k$$
$$B_{k+1} = B_k + U_k$$

In iteration (2.7)

$$U_k = U(B_k, \alpha_k s_k, g_{k+1}, g_k)$$

is usually a rank one or rank two matrix with

(2.8) $$B_{k+1} s_k = g_{k+1} - g_k \ .$$

Equation (2.8) is called the quasi-Newton equation. The advantage of iteration (2.7) over (2.4) is that the only new information required to obtain B_{k+1} from B_k is the calculation of the gradient g_{k+1}. The computational savings is that only n instead of $\frac{1}{2}n^2$ scalar function evaluations are required to obtain an approximate Hessian at step k. Moreover, the task of programming the Hessian is avoided.

The price one pays for the computational savings obtained through the use of a quasi-Newton method is that the local quadratic rate of convergence that is enjoyed by iteration (2.4) is no longer guaranteed. Instead, if the iterates $\{x_k\}$ defined by (2.7) converge to a point x* where g(x*) = 0 and G(x*) is nonsingular, then

(2.10) $$\lim_{k\to\infty} \frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} = 0$$

under suitable restrictions on $\{B_k\}$ and G. A sequence $\{x_k\}$ that satisfies (2.10) is said to converge Q-superlinearly to x*. A thorough account of iterations of type (2.7) can be found in the excellent survey by Dennis and Moré [7].

Evidently, the linear systems

$$G_k s_k = -g_k$$

that must be solved at each step are central to the implementation of these methods. Solving linear systems Ax = b using matrix factorizations costs 1/3 as much as computing A^{-1} and has been shown to be numerically more stable than computing A^{-1}. Since the linear systems arising in the context of non-linear optimization have symmetric coefficient matrices it is of great interest to obtain efficient and stable methods for factoring symmetric matrices.

The advent of quasi-Newton methods has inspired a large portion of the research in an area of numerical linear algebra called updating matrix factorizations. Since the matrix B_{k+1} in (2.7) differs from B_k by at most a rank two matrix, one might expect that the factorization of B_{k+1} could be obtained with less computational effort if the information contained in the factorization of B_k were used. This has indeed been found to be the case.

The types of quasi-Newton updating formulas that have been found to be most successful so far have satisfied

(2.11a) $\quad B_k$ symmetric => B_{k+1} symmetric,

(2.11b) $\quad B_k$ positive definite => B_{k+1} positive definite.

For this reason, there has been much work concerned with updating

variants of the Cholesky factorization [9,13,14] of a symmetric positive definite matrix. No algorithm has been given for maintaining and updating the factorization of a symmetric (possibly indefinite) matrix. However, there is at least one promising updating formula that does not satisfy (2.11b): Powell's symmetric form of Broyden's update [18].

3. The Symmetric Indefinite Decomposition

The modified Newton method that is to be presented in Chapter IV relies heavily on the factorization of a symmetric matrix given by Bunch and Parlett [5] and later improved upon by Bunch and Kaufman [4]. One would hope that the techniques developed for the modified Newton method could be extended to a quasi-Newton method. As a step towards realizing this extension, the updating problem for the symmetric indefinite factorization has been studied. A numerical method for updating the factorization of a symmetric matrix when followed by a rank one change is presented in Chapter II. A detailed error analysis of this algorithm is given in Chapter III.

As noted above, most of the work in quasi-Newton methods has been concerned with maintaining positive definite approximations to the Hessian. Hence the work in numerical linear algebra generated by these methods has been primarily concerned with updating some form of Cholesky's method for factoring a symmetric positive definite matrix.

The factorization of Bunch and Parlett does not require that the matrix be positive definite. Given any symmetric matrix $A \in R^{n\times n}$ this algorithm produces a permutation matrix Q, a unit lower triangular matrix M, and a block diagonal matrix D such that

$$QAQ^t = MDM^t .$$

The diagonal blocks of D are order one or two. If we call an arithmetic operation a multiplication followed by an addition, then the number of arithmetic operations required to obtain this decomposition is $\frac{1}{6}n^3 + O(n^2)$. (If $x = \sum_{j=1}^{k} a_j n^j$ with $a_k \neq 0$ we write $x = O(n^2)$ and say x is of order n^k.)

Another algorithm for factoring a symmetric indefinite matrix was given by Aasen [1]. In that algorithm one obtains

$$QAQ^t = LTL^t ,$$

where Q is a permutation matrix, L is unit lower triangular, and T is tridiagonal. This factorization requires $\frac{1}{6}n^3 + O(n^2)$ arithmetic operations also.

Since these factorizations both require $\frac{1}{6}n^3 + O(n^2)$ operations, an updating algorithm for obtaining the factorization of a symmetric matrix $\tilde{A} = A+U$ when the factorization of A is known should require at most $O(n^2)$ arithmetic operations. Otherwise, there would be no computational advantage over the alternative of actually computing the matrix $\tilde{A}$ and factoring the result. The updating algorithm presented in Chapter II is concerned with the following problem:

> Given $A \in R^{n\times n}$, $A = A^t$, $z \in R^n$, $\sigma \in R$, let $QAQ^t = MDM^t$ be the Bunch-Parlett factorization of A; let
>
> $$\tilde{A} = A + \sigma zz^t .$$
>
> Find an algorithm to compute
>
> $$\tilde{Q}\tilde{A}\tilde{Q}^t = \tilde{M}\tilde{D}\tilde{M}^t$$

which requires at most $O(n^2)$ arithmetic operations.

This algorithm makes use of the block structure of the matrix D. We found no similar way to take advantage of the corresponding tridiagonal matrix T in Aasen's factorization. At present we do not know of an algorithm for updating the factorization of Aasen. The updating algorithm that is presented here requires between $n^2 + O(n)$ and $\frac{11}{6}n^2 + O(n)$ operations. The method is shown to be stable as long as the factor M is well conditioned with respect to solving linear systems. These statements are made precise in chapters II and III.

4. Computational Results and Conclusions

Chapter V is concerned with presenting computational evidence in support of the theoretical work described in chapters II, III, IV. The computations were carried out at Argonne National Laboratory using an IBM 370/195. All computations were done in double-precision arithmetic.

The updating algorithm has been tested for accuracy and timing over a wide range of updating problems. We have included timings for problems of various orders. The accuracy of solutions to linear systems using the updating algorithm have been compared with solutions obtained by computing and factoring $A+\sigma zz^t$ at each step. The results are very encouraging. They indicate that the bounds obtained in our analysis are quite pessimistic and that the algorithm does not break down even when the updating process is applied over many iterations.

The unconstrained optimization algorithm was applied to many of the standard test problems which appear in the literature. Although more work is needed to obtain an algorithm that can be recommended for

general use, the initial results show this algorithm to be competitive with the algorithm of Gill and Murray [11]. In any case the underlying idea is worthy of further research. It would be of great interest if the ideas could be extended to a quasi-Newton method and to a constrained optimization algorithm.

Chapter II

Updating Factorizations of Symmetric Matrices

1. Introduction

Methods in numerical linear algebra are usually concerned with the solution of a single linear problem. For example, a particular method might be concerned with the solution of the linear system $Ax = b$ where $A \in R^{n\times n}$ and $x,b \in R^n$. Yet in practice we are often faced with solving a sequence of linear problems which are closely related. For instance, we may be interested in solving a sequence of $n\times n$ linear systems

$$(1.1) \qquad \left.\begin{aligned} A_k x_k &= b_k \\ A_{k+1} &= A_k + U_k \end{aligned}\right\} \; k=1,2,\ldots .$$

In many cases of interest U_k is of low rank. Often the rank of U_k is one or two.

Direct methods for solving the main problems of numerical linear algebra have come to rely heavily upon the use of matrix factorizations. For full matrices the price (in terms of arithmetic operations) of such factorizations is generally substantial. For instance, the relevant factorization needed to solve (1.1) requires $O(n^3)$ arithmetic operations for each A_k. However, when $U_k = A_{k+1}-A_k$ has low rank, one might expect that the factorization of A_{k+1} could be computed in an order of magnitude fewer operations using our knowledge of the factorization of A_k. For example, in (1.1) we would aim for algorithms which require only $O(n^2)$ arithmetic operations.

Here we shall be concerned with factorizations used in solving

the problem (1.1) when the matrices A_k and U_k are symmetric, and where each U_k is a rank one matrix. Then (1.1) has the form

$$\left.\begin{aligned} A_k x_k &= b_k \\ A_{k+1} &= A_k + \sigma_k z_k z_k^t \end{aligned}\right\} k=1,2,\ldots \tag{1.2}$$

where each $z_k \in R^n$, $\sigma_k \in R$, $A_k = A_k^t$. This problem arises for instance in quasi-Newton methods for optimization problems [7].

Thus we shall concern ourselves with obtaining the factorization of

$$\tilde{A} = A + \sigma z z^t \tag{1.3}$$

not by forming $\tilde{A}$ explicitly, but by using the factorization of A. Such a process is called updating a matrix factorization.

There are two important and very distinct cases:

(i) A is positive definite,

(ii) A is indefinite.

In case (i) A may be factored in a numerically stable way into

$$A = LDL^t ,$$

where $L \in R^{n\times n}$ is a unit lower triangular matrix, and $D \in R^{n\times n}$ is a diagonal matrix with positive diagonal elements. No pivoting is required to obtain numerical stability in the positive definite case. However, in case (ii) such a factorization may not even exist. For example consider the matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} .$$

A numerically stable method for obtaining a factorization of A in case

(ii) is given in [5] by Bunch and Parlett, and is later revised in [4] by Bunch and Kaufman. By this method one obtains a permutation matrix Q, a lower triangular matrix M, and a block diagonal matrix D such that

$$QAQ^t = MDM^t \quad . \tag{1.4}$$

The diagonal blocks of D are order one or two. Whenever $D_{i+1,i} \neq 0$ then $M_{i+1,i} = 0$. Also, $M_{ii} = 1$ for all i.

The case in (1.3) where both A and $\tilde{A}$ are theoretically known to be positive definite has been studied and updating algorithms are given in [9,13,14]. The case where A and $\tilde{A}$ are symmetric but possibly indefinite has not been studied.

In the following sections we shall present and analyze an algorithm for computing $\tilde{Q}$, $\tilde{M}$, $\tilde{D}$, when given the factorization (1.4), such that

$$\tilde{Q}\tilde{A}\tilde{Q}^t = \tilde{M}\tilde{D}\tilde{M}^t \quad , \tag{1.5}$$

where $\tilde{A}$ is given by (1.3). The algorithm requires between $n^2 + 4n$ and $\frac{11}{6}n^2 + \frac{55}{6}n + \frac{25}{3}$ arithmetic operations and at most 2n comparisons. Here an arithmetic operation is considered to be a floating point multiplication followed by an addition. Divisions are counted as multiplications. The operation count compares favorably with the alternative of computing $A + \sigma zz^t$ and then factoring this matrix into $\tilde{M}\tilde{D}\tilde{M}^t$. This would require $\frac{1}{2}n^2 + n$ multiplications together with $\frac{1}{2}n^2$ additions to form the new matrix. It would then require at most

$$\frac{1}{6}n^3 + \frac{3}{4}n^2 + \frac{1}{3}n$$

operations to obtain the new decomposition. Therefore, a total of at most

$$\frac{1}{6}n^3 + \frac{5}{4}n^2 + \frac{4}{3}n$$

operations would be needed.

Thus it is advantageous to use the updating algorithm whenever $n \geq 10$. However, it should be emphasized that the upper bound on the number of operations required by the updating algorithm is a worst case bound. Computational results indicate that the worst case seldom occurs. Therefore, we expect that in practice the crossover number would be much smaller.

2. Description of the Algorithm

We shall begin by describing a basic algorithm with no pivoting. The algorithms given in [9,14] for the positive definite case will be presented as modifications to this basic algorithm. The modifications were designed to insure numerical stability. The algorithm we present for the indefinite case is also a modification of this basic algorithm. However, it is necessarily more complicated since the pivoting must be updated.

Assume for the moment that no permutations were required to obtain

$$A = MDM^t$$

with M (block) unit lower triangular, and D block diagonal with one-by-one or two-by-two diagonal blocks. Then we may write

$$A = \sum_{j=1}^{m} M_j D_j M_j^t ,$$

where the D_j are the diagonal blocks of D and the M_j are the block

columns of M. Let

$$\tilde{A} = A + \sigma z z^t ,$$

and let $Mp = z$. Denote

$$A^{(k)} = \sum_{j=k}^{m} M_j D_j M_j^t, \quad w^{(k)} = \sum_{j=k}^{m} M_j p_j ,$$

where $p^t = (p_1^t, p_2^t, \ldots, p_m^t)$.

Suppose that $\tilde{D}_1 \equiv D_1 + \sigma p_1 p_1^t$ is non-singular, and let $\tilde{D}_1 b_1 = \sigma p_1$. Then take $\tilde{M}_1 = M_1 + w^{(2)} b_1^t$. Note that only the elements below the identity part of M_1 are altered;

$$\tilde{M}_1 = \begin{bmatrix} I \\ x \\ x \\ x \end{bmatrix} + \begin{bmatrix} 0 \\ y \\ y \\ y \end{bmatrix} ,$$

where the x's and y's denote possibly non-zero quantities. We have that

(2.1)
$$\begin{aligned} \tilde{A} &= M_1(D_1 + \sigma p_1 p_1^t)M_1^t + \sigma(M_1 p_1 w^{(2)t} + w^{(2)} p_1^t M_1^t) \\ &\quad + A^{(2)} + \sigma w^{(2)} w^{(2)t} \\ &= (M_1 + w^{(2)} b_1^t)\tilde{D}_1(M_1 + w^{(2)} b_1^t)^t \\ &\quad + \sigma(M_1 p_1 w^{(2)t} + w^{(2)} p_1^t M_1^t) \\ &\quad - (M_1 \tilde{D}_1 b_1 w^{(2)t} + w^{(2)} b_1^t \tilde{D}_1 M_1^t) \\ &\quad + (\sigma - b_1^t \tilde{D}_1 b_1) w^{(2)} w^{(2)t} \\ &\quad + A^{(2)} \\ &= \tilde{M}_1 \tilde{D}_1 \tilde{M}_1^t + A^{(2)} + \sigma' w^{(2)} w^{(2)t} . \end{aligned}$$

Observe that the matrix $A^{(2)} + \sigma' w^{(2)} w^{(2)t}$ has the form

$$\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & X \end{array}\right]$$

and thus we may recursively apply this procedure to obtain

$$A = \sum_{j=1}^{m} \tilde{M}_j \tilde{D}_j \tilde{M}_j^t$$

as long as $\tilde{D}_j \equiv D_j + \sigma_j p_j p_j^t$ is non-singular for $1 \leq j \leq m$. This assumption on $\tilde{D}_j$ is theoretically always satisfied in the positive definite case. However, this cannot be guaranteed in the indefinite case. After establishing some preliminary results concerning these computations we shall discuss some of the numerical algorithms that have been proposed for the positive definite case.

Lemma (2.1). Let D and $D + \sigma pp^t$ be non-singular. Then the solution to

$$\text{(i)} \quad (D + \sigma pp^t)b = \sigma p$$

is given by $b = \theta D^{-1}p$, where $\theta = \sigma/(1 + \sigma p^t D^{-1}p)$. Moreover,

$$\text{(ii)} \quad \det(D + \sigma pp^t) = \det D(1 + \sigma p^t D^{-1}p) \text{ and the updated } \sigma' = \sigma - b^t \tilde{D} b$$

is given by

$$\text{(iii)} \quad \sigma' = \frac{\sigma}{1 + \sigma p^t D^{-1} p}.$$

Proof:

(i) follows by substitution,

(ii) Sherman-Morrison formula (or direct computation),

(iii) follows by substitution.

□

Thus if all the $\tilde{D}_j$ (for $1 \le j \le m$) are non-singular we have the formula

$$\sigma_{j+1} = \frac{\sigma_j}{1 + \sigma_j p_j^t D_j^{-1} p_j} \quad \text{for} \quad 1 \le j \le m ,$$

and

$$\frac{\sigma_1}{\sigma_k} = \frac{\prod_{j=1}^{k} \det \tilde{D}_j}{\prod_{j=1}^{k} \det D_j} ,$$

hence

$$\frac{\sigma_1}{\sigma_{m+1}} = \frac{\det \tilde{A}}{\det A} .$$

In the case that both A and $\tilde{A}$ are positive definite, these formulas point out the necessity of maintaining σ_j with the same sign as σ. We note also that we may recursively compute $t_j = \sigma_j^{-1}$ as follows:

$$t_{j+1} = t_j + p_j^t D_j^{-1} p_j , \tag{2.2}$$

and we have the relation

$$\frac{t_{j+1}}{t_j} = \frac{\det \tilde{D}_j}{\det D_j} . \tag{2.3}$$

When A is positive definite, $\tilde{d}_j = \det \tilde{D}_j$, $d_j = \det D_j$ and $D = \mathrm{diag}(d_1, \ldots, d_n)$.

Now, often in practice one knows theoretically that the matrix $\tilde{A}$ should be positive definite when A is positive definite. In the case that σ is positive there is no difficulty since the recursion for the t_j's yields an increasing sequence, and $\tilde{d}_j = \frac{t_{j+1}}{t_j} d_j$. Thus the $\tilde{d}_j$ are all positive and $\tilde{d}_j \ge d_j$. The following algorithm results for $\sigma > 0$:

(2.4) $t_1 = \sigma^{-1}, \quad w^{(1)} = z, \quad A = MDM^t$

for i = 1 step 1 until n do

(1) $p_i = w_i^{(i)}$

(2) $t_{i+1} = t_i + p_i^2/d_i$

(3) $\tilde{d}_i = d_i(t_{i+1}/t_i)$

(4) terminate if i = n

(5) $b_i = (p_i/d_i)/t_{i+1}$

(6) $w^{(i+1)} = w^{(i)} - p_i M_i$

(7) $\tilde{M}_i = M_i + b_i w^{(i+1)}$.

Note that the number of arithmetic operations required is $n^2 + O(n)$, since only i operations are needed at steps 6 and 7.

Difficulties arise when $\sigma < 0$ because round off error may cause a t_{i+1} to be positive, and hence $\tilde{d}_i$ will be negative indicating that the computed $\tilde{A}$ is not positive definite.

Two remedies have been proposed. One of these [9] is to compute the vector p such the Mp = z at the outset. It is noted that in the application to quasi-Newton methods, the vector p is often available anyway. If $\sigma < 0$ then calculate the t_j, for $2 \leq j \leq n+1$ from the formula (2.2). If one of the t_j should turn out to be positive then the t_j are recalculated using

(2.5) $t_{n+1} = \varepsilon/\sigma$,

$t_j = t_{j+1} - p_j^2/d_j, \quad j=n,n-1,\ldots,1$

where ε is the relative machine precision. These new values of t_i are then used in place of the old ones in (2.4), steps 3 through 7. The effect is to replace σ by t_1^{-1} which gives a problem that is close to the

original problem, and for which the computed $\tilde{A}$ will be positive definite.

In [14] another approach is taken which yields a similar algorithm. The major differences being that t_{n+1} is set to ε if some t_j is positive and a backwards recurrence formula is used to compute $\tilde{M}$. Thus, in place of (2.4) steps 6 and 7, we would have

(2.6) $w^{(n+1)} \equiv 0$

for i = n step -1 until 1 do

(1) $w_i^{(i)} = p_i$

(2) $M_i = M_i + b_i w^{(i+1)}$

(3) $w^{(i)} = w^{(i+1)} + p_i M_i$.

However, there seems to be the need for additional storage in (2.6). Note that the computation of $w^{(i)}$ requires knowledge of M_i which has presumably been overwritten at step (2) of (2.6).

In [9] an error analysis of this process has been given. That analysis shows that

$$MDM^t = A + zz^t + E ,$$

where the elements of E have first order terms in ε which depend on the ratio $\sqrt{\tilde{d}_i/d_i}$ in (2.4) step 7 is used. However, it is possible to show that

(2.7) $$\tilde{M}_i = M_i(d_i/\tilde{d}_i) + b_i w^{(i)}$$

and here the error terms depend on the ratio $\sqrt{d_i/\tilde{d}_i}$ which is less than 1 when $\sigma > 0$. In both [9] and [14] one switches to (2.7) only if the ratio $\sqrt{\tilde{d}_i/d_i}$ becomes larger than some bound.

This leads us to the following algorithm which is a slight

modification of the composite t-method given in [9].

(2.8)

(1) terminate if $\sigma = 0$; put $t_1 = \sigma^{-1}$ and $w^{(1)} = z$;

(2) if $\sigma > 0$ go to 6;

(3) if p is not available solve $Mp = z$ for p;

(4) for i=1,2,...,n do $t_{i+1} = t_i + p_i^2/d_i$;

(5) if any $t_i \geq 0$ then

begin

$t_{n+1} = \varepsilon/\sigma$;

for i=n,n-1,...,1 do $t_i = t_{i+1} - p_i^2/d_i$;

end

(6) for i = 1 step 1 until n do

if $\sigma > 0$ then begin $p_i = w_i^{(i)}$; $t_{i+1} = t_i + p_i^2/d_i$; end;

$\theta_i = t_{i+1}/t_i$; $\tilde{d}_i = d_i\theta_i$;

terminate if i = n

$b_i = (p_i/d_i)/t_{i+1}$;

if $\theta_i > 4$ then

begin

$\gamma_i = t_i/t_{i+1}$;

$\tilde{M}_i = \gamma_i M_i + b_i w^{(i)}$;

$w^{(i+1)} = w^{(i)} - p_i M_i$;

end

else

begin

$w^{(i+1)} = w^{(i)} - p_i M_i$;

$\tilde{M}_i = M_i + b_i w^{(i+1)}$;

end .

The situation becomes completely different when the matrices A and $\tilde{A}$ are not assumed to be positive definite. In order to obtain a stable algorithm for solving Ax = b pivoting must be used to factor A [3,5] and we obtain

$$QAQ^t = MDM^t .$$

Moreover, the following example shows that $\tilde{D}_i$ in (2.1) may be singular even though both A and $\tilde{A}$ are non-singular.

Let $A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & \frac{1}{4} \end{pmatrix}$ $(= MDM^t$, where M = I and D = A),

let $\sigma = \frac{1}{2}$, $z = (1,-1,1)^t$.

Then $\tilde{D}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 \\ -1 \end{pmatrix}[1,-1] = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$ is singular but $\tilde{A} = A + \sigma zz^t$ satisfies $\det \tilde{A} = -\frac{1}{2}$. Therefore, some pivoting strategy must be employed to avoid the breakdown of the computation (2.1). The main difficulty in updating the pivoting strategy is maintaining $\tilde{M}$ in triangular form.

We shall now describe the pivoting strategy given in [5] for the Bunch-Parlett factorization in some detail. This strategy will be used in a portion of the updating algorithm, so we include its description for the sake of completeness.

Given a symmetric non-singular matrix A with elements a_{ij} the factorization proceeds as follows:

Let $0 < \alpha < 1$ be fixed.

Let $\nu = \max_{1 \le i \le n} |a_{ii}|$ and let $\mu = \max_{i \ne j} |a_{ij}|$.

If $\nu \geq \alpha\mu$, let i be the smallest index such that $|a_{ii}| = \nu$. Let Q_1 be the identity matrix with rows 1 and i interchanged. Then the matrix $Q_1 A Q_1^t$ has the element a_{ii} in the (1,1) position. The first step of the factorization is to write

$$Q_1 A Q_1^t = \left(\begin{array}{c|c} \delta_1 & v^t \\ \hline v & A' \end{array}\right)$$

$$= \begin{pmatrix} 1 & 0 \\ v\delta_1^{-1} & I \end{pmatrix} \left(\begin{array}{c|c} \delta_1 & 0 \\ \hline 0 & A'-\delta_1^{-1}vv^t \end{array}\right) \begin{pmatrix} 1 & 0 \\ v\delta_1^{-1} & I \end{pmatrix}^t .$$

Thus

$$M_1^{-1} Q_1 A Q_1^t M_1^{-t} = \left(\begin{array}{c|c} \delta_1 & 0 \\ \hline 0 & A^{(2)} \end{array}\right) ,$$

where $M_1 = \begin{pmatrix} 1 & 0 \\ v\delta_1^{-1} & I \end{pmatrix}$, and $A^{(2)} = A' - \delta_1^{-1}vv^t$. If $\nu < \alpha\mu$, let $i > j$ indices such that $|a_{ij}| = \mu$. Let Q_1 be the identity matrix with row i interchanged with row 2 and row j interchanged with row 1. Then the matrix $Q_1 A Q_1^t$ has the element a_{ij} in the (2,1) position. In this case the first step of the factorization is to write

$$Q_1 A Q_1^t = \left(\begin{array}{c|c} D_1 & V^t \\ \hline V & A' \end{array}\right)$$

$$= \begin{pmatrix} I & 0 \\ VD_1^{-1} & I \end{pmatrix} \left(\begin{array}{c|c} D_1 & \\ \hline 0 & A'-VD_1^{-1}V^t \end{array}\right) \begin{pmatrix} I & 0 \\ VD_1^{-1} & I \end{pmatrix}^t .$$

Here V is the first two columns of $Q_1 A Q_1^t$ below the (2,1) and (2,2) positions, and D_1 is a two-by-two matrix. Also, $\det D_1 = a_{ii}a_{jj} - a_{ij}^2 \leq (\alpha^2-1)\mu^2 < 0$. Thus

$$M_1^{-1} Q_1 A Q_1^t M_1^{-t} = \left(\begin{array}{c|c} D_1 & 0 \\ \hline 0 & A^{(2)} \end{array}\right) ,$$

where $M_1 = \begin{bmatrix} I & 0 \\ VD_1^{-1} & I \end{bmatrix}$, and $A^{(2)} = A' - VD_1^{-1}V^t$.

The factorization now proceeds by applying the same pivoting strategy to the reduced matrix $A^{(2)}$. The end result is that

$$M_k^{-1}Q_kM_{k-1}^{-1}Q_{k-1} \cdots M_1^{-1}Q_1AQ_1^tM_1^{-t} \cdots Q_{k-1}^tM_{k-1}^{-t}Q_k^tM_k^{-t} = D ,$$

where D is a block diagonal matrix with 1×1 or 2×2 diagonal blocks. Hence,

$$A = Q_1M_1Q_2M_2 \cdots Q_kM_kDM_k^tQ_k^t \cdots M_2^tQ_2^tM_1^tQ_1^t .$$

Since $Q_i^{-1} = Q_i^t$ for $1 \leq i \leq k$ we may write

$$Q_1M_1Q_2M_2 \cdots Q_kM_k = Q^t\bar{M}_1\bar{M}_2 \cdots \bar{M}_k ,$$

where $Q^t = Q_1Q_2 \cdots Q_k$,

and $\bar{M}_j = Q_{j+1}^tQ_{j+2}^t \cdots Q_k^tM_jQ_k \cdots Q_{j+2}Q_{j+1}$.

Then $\bar{M}_j$ has the same form as M_j and thus if we take

$$M = \bar{M}_1\bar{M}_2 \cdots \bar{M}_k$$

then M is a block unit lower triangular matrix such that

$$QAQ^t = MDM^t .$$

For fixed α, $0 < \alpha < 1$, the strategy just described shall be called the diagonal pivoting strategy S_α. When α is chosen to be $(1+\sqrt{17})/8$, the factorization is almost as stable as Gaussian elimination with complete pivoting [3,5]. A modification of this strategy, which is comparable to Gaussian elimination with partial pivoting, is given in [4]. The algorithm in [5] requires between $\frac{1}{12}n^3 + \frac{3}{8}n^2 + \frac{1}{6}n$ and $\frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n$ comparisons, while the algorithm in [4] requires at most n^2-1 comparisons.

Now, in order to establish the theorem that we shall use to construct the algorithm for updating this factorization shall need some preliminary lemmas.

Lemma (2.2). Let $A \in R^{n\times n}$ be symmetric with eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and let $\tilde{A} = A + \sigma zz^t$ for some $z \in R^n$, $\sigma \in R$. If $\sigma > 0$ then $\tilde{A}$ has eigenvalues $\tilde{\lambda}_i$ such that

$$\lambda_1 \leq \tilde{\lambda}_1 \leq \lambda_2 \leq \tilde{\lambda}_2 \leq \ldots \leq \lambda_n \leq \tilde{\lambda}_n ,$$

while if $\sigma \leq 0$ then the eigenvalues of $\tilde{A}$ can be arranged so that

$$\tilde{\lambda}_1 \leq \lambda_1 \leq \tilde{\lambda}_2 \leq \ldots \leq \tilde{\lambda}_n \leq \lambda_n .$$

Proof: [20] pp. 95-98. □

Remark: In particular if A is non-singular then at most one of the $\tilde{\lambda}_i$ is zero.

With Lemma (2.2) and the pivoting strategy just described we can establish

Lemma (2.3). Let $V = \begin{pmatrix} I \\ V' \end{pmatrix}$, where $I \in R^{\ell\times\ell}$ and $V' \in R^{k\times\ell}$. Suppose that $D = D^t \in R^{\ell\times\ell}$ is non-singular and that $w \in R^{k+\ell}$, $\sigma \in R$. Define

$$C \equiv VDV^t + \sigma ww^t . \tag{2.9}$$

Then there is an $\ell\times\ell$ permutation matrix Q such that

$$\begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix} C \begin{pmatrix} Q^t & 0 \\ 0 & I \end{pmatrix} = \tilde{V}\tilde{D}\tilde{V}^t + \begin{cases} (v,\bar{w})B(v,\bar{w})^t, \\ \bar{w}\bar{w}^t \end{cases} \tag{2.10}$$

where

(i) $\tilde{D}$ is a non-singular block diagonal matrix with 1×1 or 2×2 diagonal blocks,

(ii) $\tilde{V}$ is block unit lower trapezoidal,

(iii) $v = \begin{pmatrix} 0 \\ 1 \\ \bar{v} \end{pmatrix}$, $\bar{v} \in R^k$,

(iv) $\bar{w} = \begin{pmatrix} 0 \\ 0 \\ \bar{\bar{w}} \end{pmatrix}$, $\bar{\bar{w}} \in R^k$, and

(v) $B \in R^{2\times 2}$.

Proof: Write

$$C = (V,w)\begin{pmatrix} D & 0 \\ 0 & \sigma \end{pmatrix}(V,w)^t \tag{2.11}$$

$$= \begin{pmatrix} I & 0 \\ V' & \frac{1}{\mu} w' \end{pmatrix}\begin{pmatrix} \bar{D} & \mu s \\ \mu s^t & \mu^2\sigma \end{pmatrix}\begin{pmatrix} I & 0 \\ V' & \frac{1}{\mu} w' \end{pmatrix}^t$$

where

$$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix},\ w_1 \in R^{\ell},\ w' = w_2 - V'w_1,$$

$$\bar{D} = D + \sigma w_1 w_1^t,$$

and $$s = \sigma w_1 .$$

Here μ may be any positive real number; if μ is chosen small enough then the diagonal pivoting strategy S_α will give either

$$\begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \bar{D} & \mu s \\ \mu s & \mu^2\sigma \end{pmatrix}\begin{pmatrix} Q^t & 0 \\ 0 & 1 \end{pmatrix} \tag{2.12}$$

$$= \begin{pmatrix} \bar{M} & 0 \\ \mu b^t & 1 \end{pmatrix}\begin{pmatrix} \tilde{D} & 0 \\ 0 & \mu^2\sigma' \end{pmatrix}\begin{pmatrix} \bar{M} & 0 \\ \mu b^t & 1 \end{pmatrix}^t$$

or

$$= \begin{pmatrix} \bar{M} & 0 & 0 \\ \bar{m}^t & 1 & 0 \\ \mu b^t & 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{D} & & 0 & 0 \\ & & 0 & 0 \\ 0 & 0 & \delta & \mu\beta \\ 0 & 0 & \mu\beta & \mu^2\sigma' \end{pmatrix} \begin{pmatrix} \bar{M} & 0 & 0 \\ \bar{m}^t & 1 & 0 \\ \mu b^t & 0 & 1 \end{pmatrix}^t ,$$

where D is block diagonal with 1×1 or 2×2 diagonal blocks, and $\bar{M}$ has the corresponding block unit lower triangular structure. Since D is non-singular, Lemma (2.2) implies that $\bar{D}$ has at most one zero eigenvalue. The diagonal pivoting strategy preserves the inertia of $\bar{D}$. Therefore, $\tilde{D}$ is non-singular. Note that $\delta = 0$ in (2.13) if and only if $\bar{D}$ is singular and in this case we cannot carry the decomposition further without permuting the last row and column of

$$\begin{pmatrix} \bar{D} & \mu s \\ \mu s^t & \mu^2\sigma \end{pmatrix} .$$

If (2.12) is obtained then

$$\begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix} C \begin{pmatrix} Q^t & 0 \\ 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} Q^t & 0 \\ V'Q^t & \frac{1}{\mu} w' \end{pmatrix} \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{D} & \mu s \\ \mu s^t & \mu^2\sigma \end{pmatrix} \begin{pmatrix} Q^t & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} Q^t & \\ V'Q^t & \frac{1}{\mu} w' \end{pmatrix}^t \begin{pmatrix} Q^t & 0 \\ 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} I & 0 \\ \hat{V} & \frac{1}{\mu} w' \end{pmatrix} \begin{pmatrix} \bar{M} & 0 \\ \mu b^t & 1 \end{pmatrix} \begin{pmatrix} \tilde{D} & 0 \\ 0 & \mu^2\sigma' \end{pmatrix} \begin{pmatrix} \bar{M} & 0 \\ \mu b^t & 0 \end{pmatrix}^t \begin{pmatrix} I & 0 \\ \hat{V} & \frac{1}{\mu} w' \end{pmatrix}^t$$

$$= \left(\begin{array}{cc|c} \bar{M} & & 0 \\ \hat{\bar{V}}\bar{M} + w'b^t & & w' \end{array}\right) \begin{pmatrix} \tilde{D} & 0 \\ 0 & \sigma' \end{pmatrix} \left(\begin{array}{cc|c} \bar{M} & & 0 \\ \hat{\bar{V}}\bar{M} + w'b^t & & w' \end{array}\right)^t , \quad \text{where}$$

$$\begin{pmatrix} I & 0 \\ \hat{V} & 0 \end{pmatrix} = \begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} Q^t & \\ V' & Q^t \end{pmatrix} = \begin{pmatrix} I & \\ V' & Q^t \end{pmatrix} .$$

Here we take

$$\tilde{V} = \begin{pmatrix} \bar{M} \\ \hat{V}M + w'b^t \end{pmatrix}, \text{ and } \bar{\bar{w}} = w'.$$

If (2.13) is obtained then

$$\begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix} C \begin{pmatrix} Q^t & 0 \\ 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} Q^t & 0 \\ V'Q^t & \frac{1}{\mu} w' \end{pmatrix} \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{D} & \mu s \\ \mu s^t & \mu^2 \sigma \end{pmatrix} \begin{pmatrix} Q^t & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} Q^t & 0 \\ V'Q^t & \frac{1}{\mu} w' \end{pmatrix}^t \begin{pmatrix} Q^t & 0 \\ 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} I & 0 & 0 \\ \hat{V} & v & \frac{1}{\mu} w' \end{pmatrix} \begin{pmatrix} \bar{M} & 0 & 0 \\ \bar{m}^t & 1 & 0 \\ \mu b^t & 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{D} & 0 & 0 \\ & 0 & 0 \\ 0 \; 0 & \delta & \mu\beta \\ 0 \; 0 & \mu\beta & \mu^2\sigma' \end{pmatrix} \begin{pmatrix} \bar{M} & 0 & 0 \\ \bar{m}^t & 1 & 0 \\ \mu b^t & 0 & 1 \end{pmatrix}^t \begin{pmatrix} I & 0 & 0 \\ \hat{V} & v & \frac{1}{\mu} w' \end{pmatrix}^t$$

$$= \left(\begin{array}{c|c|c} \bar{M} & 0 & 0 \\ \hat{V}\bar{M} + v\bar{m}^t + w'b^t & v & w' \end{array}\right) \begin{pmatrix} \tilde{D} & 0 & 0 \\ & 0 & 0 \\ 0 \; 0 & \delta & \beta \\ 0 \; 0 & \beta & \sigma' \end{pmatrix} \left(\begin{array}{c|c|c} \bar{M} & 0 & 0 \\ \hat{V}\bar{M} + v\bar{m}^t + w'b^t & v & w' \end{array}\right)^t$$

where $\begin{pmatrix} I & 0 \\ \hat{V} & v \end{pmatrix} = \begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} Q^t \\ V'Q^t \end{pmatrix} = \begin{pmatrix} I \\ V'Q^t \end{pmatrix}$.

Here $v = \begin{pmatrix} 1 \\ \bar{v} \end{pmatrix}$, where $\bar{v} \in R^k$,

and we take

$$\tilde{V} = \begin{pmatrix} \bar{M} \\ \hat{V}\bar{M} + v\bar{m}^t + w'b^t \end{pmatrix}, \quad B = \begin{pmatrix} \delta & \beta \\ \beta & \sigma' \end{pmatrix}, \; \bar{\bar{w}} = w'.$$

This gives the desired result. □

Observe that the scale factor μ does not actually enter into the

computations and thus explicit scaling need not be implemented in a code. Also, we note that for the intended application, we will have $\ell \leq 3$ in Lemma (2.3). When $\ell \leq 3$ we have

$$C = (V,w)\begin{pmatrix} D & 0 \\ 0 & \sigma \end{pmatrix}(V,w)^t$$

with D of order at most 3. Then in the computations non-singular matrices of order at most 4 are inserted between the factors on the right and permutations are used to obtain

$$\begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix} C \begin{pmatrix} Q^t & 0 \\ 0 & I \end{pmatrix} = (\tilde{V}|v,\bar{w})\begin{pmatrix} \tilde{D} & 0 \\ 0 & B \end{pmatrix}(\tilde{V}|v,\bar{w})^t \ .$$

Thus, some fixed number of arithmetic operations are required to compute $\tilde{D}$ and B. Also, some fixed multiple of k arithmetic operations are required to compute $\tilde{V}$.

Before we give the main theorem of this section we shall need to establish one more lemma. The proof of the lemma is trivial but it is included for the sake of clarity in some of the following computations.

Lemma (2.4). Let $B = \begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix} \in R^{2\times 2}$, where $\beta \neq 0$. Then

$$B = \begin{pmatrix} 1 & -\gamma \\ \gamma & 1 \end{pmatrix}\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} 1 & \gamma \\ -\gamma & 1 \end{pmatrix},$$

with $|\lambda_1| \geq |\lambda_2|$.

Proof: Let μ_1,μ_2 be the eigenvalues of B with $|\mu_1| \geq |\mu_2|$. Since $\beta \neq 0$, B has an eigenvector corresponding to μ_1 of the form $\binom{1}{\gamma}$. Thus

$$\begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix}\begin{pmatrix} 1 \\ \gamma \end{pmatrix} = \mu_1 \begin{pmatrix} 1 \\ \gamma \end{pmatrix}.$$

Therefore, $\delta+\beta\gamma = \mu_1$ and since $\beta \neq 0$

we have $\gamma = (\mu_1-\delta)/\beta$.

Since B is symmetric it has an orthonormal system of eigenvectors. Thus

$$B = \begin{pmatrix} 1 & -\gamma \\ \gamma & 1 \end{pmatrix}\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} 1 & \gamma \\ -\gamma & 1 \end{pmatrix},$$

where $\lambda_i = \mu_i/(1+\gamma^2)$, i=1,2. □

The following theorem will show how Lemma (2.3) can be used to obtain an $O(n^2)$ updating algorithm.

Theorem (2.1). Let $A \in R^{n\times n}$ be non-singular with $QAQ^t = MDM^t$. Suppose that $z \in R^n$, $\sigma \in R$ are such that

$$\tilde{A} = A + \sigma zz^t$$

is also non-singular. Then $\tilde{Q}\tilde{A}\tilde{Q}^t = \tilde{M}\tilde{D}\tilde{M}^t$ can be computed from the factorization of A in $O(n^2)$ arithmetic operations.

Proof: Let $w = Qz$. Then $Q\tilde{A}Q^t = MDM^t + \sigma ww^t$. We denote $\tilde{A}^{(k)} = \sum_{j=1}^{k} \tilde{M}_j\tilde{D}_j\tilde{M}_j^t$, $A^{(k)} = \sum_{j=k}^{m} M_jD_jM_j^t$. First we may write

$$Q\tilde{A}Q^t = \begin{pmatrix} I & w_1 \\ V & w_2 \end{pmatrix}\begin{pmatrix} D & 0 \\ 0 & \sigma \end{pmatrix}\begin{pmatrix} I & w_1 \\ V & w_2 \end{pmatrix}^t + A^{(2)} \equiv C^{(1)} + A^{(2)}$$

with $M_1 = \begin{pmatrix} I \\ V \end{pmatrix}$, $w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$, $D = D_1$. Then by Lemma (2.3) we may construct a permutation U_1 such that

$$(2.14)\quad \begin{pmatrix} U_1 & 0 \\ 0 & I \end{pmatrix} C^{(1)} \begin{pmatrix} U_1^t & 0 \\ 0 & I \end{pmatrix} = \begin{cases} \text{(i)} \quad \begin{pmatrix} 1 & 0 \\ v_1 & w \end{pmatrix} B \begin{pmatrix} 1 & 0 \\ v_1 & w' \end{pmatrix}^t , \\ \text{or} \\ \text{(ii)} \quad \begin{pmatrix} 1 \\ v_2 \end{pmatrix} (\tilde{\delta}) \begin{pmatrix} 1 \\ v_2 \end{pmatrix}^t + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v_1 & w' \end{pmatrix} B \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v_1 & w' \end{pmatrix}^t , \\ \text{or} \\ \text{(iii)} \quad \begin{pmatrix} I \\ V_1 \end{pmatrix} \tilde{D}_1 \begin{pmatrix} I \\ V_1 \end{pmatrix}^t + \sigma' \begin{pmatrix} 0 \\ w' \end{pmatrix} (0, w'^t) , \end{cases}$$

where $\tilde{\delta} \neq 0$, $\tilde{D}_1 \in R^{2\times 2}$ is non-singular, $B \in R^{2\times 2}$, and $\sigma' \in R$. Observe also that $\begin{pmatrix} U_1 & 0 \\ 0 & I \end{pmatrix} A^{(2)} \begin{pmatrix} U_1^t & 0 \\ 0 & I \end{pmatrix} = A^{(2)}$. If (iii) is achieved then the problem becomes

$$Q_1 \tilde{A} Q_1^t = \tilde{A}^{(1)} + A^{(2)} + \sigma' \begin{pmatrix} 0 \\ w' \end{pmatrix} (0, w'^t) ,$$

where $Q_1 = \begin{pmatrix} U_1 & 0 \\ 0 & I \end{pmatrix} Q$. Note that

$$A^{(2)} + \sigma' \begin{pmatrix} 0 \\ w' \end{pmatrix} (0, w'^t)$$

has the same form as the original problem but the dimension of the problem is decreased to n-1 or n-2.

In the following discussion we shall drop the primes and subscripts from the expressions on the right of (2.14). Also, some of the qualities appearing in (2.14) are redefined below.

If (i) holds in (2.14), then

$$Q_1\tilde{A}Q_1^t = \begin{pmatrix} 1 & 0 \\ v & w \end{pmatrix} B \begin{pmatrix} 1 & 0 \\ v & w \end{pmatrix} + A^{(2)} ,$$

while if (ii) holds then

$$Q_1\tilde{A}Q_1^t = \tilde{A}^{(1)} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v & w \end{pmatrix} B \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v & w \end{pmatrix}^t + A^{(2)} .$$

Let $\begin{pmatrix} 0 \\ I \\ V \end{pmatrix} = M_2$. Then we may write

$$Q_1AQ_1^t = \begin{cases} \text{(i)} & C^{(2)} + A^{(3)}, \\ \text{or} & \\ \text{(ii)} & \tilde{A}^{(1)} + C^{(2)} + A^{(3)} . \end{cases}$$

In (i) we have

$$C^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ \bar{v}_1 & \bar{w}_1 & I \\ \bar{v}_2 & \bar{w}_2 & V \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ \bar{v}_1 & \bar{w}_1 & I \\ \bar{v}_2 & \bar{w}_2 & V \end{pmatrix}, \tag{2.15}$$

and a similar expression in (ii). Here $v = \begin{pmatrix} \bar{v}_1 \\ \bar{v}_2 \end{pmatrix}$ and $w = \begin{pmatrix} \bar{w}_1 \\ \bar{w}_2 \end{pmatrix}$ have been partitioned so that $V\bar{v}_1$ and $V\bar{w}_1$ are defined.

Now if (i) or (ii) occurred in (2.14) then

$$B = \begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix}$$

satisfies $|\delta| < \alpha|\beta|$. Hence by Lemma (2.4)

$$B = \begin{pmatrix} 1 & -\gamma \\ \gamma & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} 1 & \gamma \\ -\gamma & 1 \end{pmatrix}$$

with $|\lambda_1| \geq |\lambda_2|$. Moreover, $\lambda_1 \neq 0$ since $\lambda_1 = 0$ implies that $B \equiv 0$.

Let

$$\begin{pmatrix} 1 & w_0 \\ v_1 & w_1 \\ v_2 & w_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \bar{v}_1 & \bar{w}_1 \\ \bar{v}_2 & \bar{w}_2 \end{pmatrix} \begin{pmatrix} 1 & -\gamma \\ \gamma & 1 \end{pmatrix}.$$

Using this expression in (2.15) gives

$$C^{(2)} = \begin{pmatrix} 1 & 0 & w_0 \\ v_1 & I & w_1 \\ v_2 & V & w_2 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & w_0 \\ v_1 & I & w_1 \\ v_2 & V & w_2 \end{pmatrix}^t \tag{2.16}$$

$$= \left(\begin{array}{cc|c} 1 & 0 & w_0 \\ 0 & I & w_1 \\ \hline \tilde{v}_2 & V & w_2 \end{array}\right) \begin{pmatrix} \hat{D} & 0 \\ 0 & \lambda_2 \end{pmatrix} \left(\begin{array}{cc|c} 1 & 0 & w_0 \\ 0 & I & w_1 \\ \hline \tilde{v}_2 & V & w_2 \end{array}\right)^t$$

with $\hat{D} = \begin{pmatrix} 1 & 0 \\ v_1 & I \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ v_1 & I \end{pmatrix}^t$ non-singular and $\tilde{v}_2 = v_2 - Vv_1$. Now, Lemma (2.3) may be applied to obtain

$$\begin{pmatrix} U_2 & 0 \\ 0 & I \end{pmatrix} C^{(2)} \begin{pmatrix} U_2^t & 0 \\ 0 & I \end{pmatrix} = \begin{pmatrix} I \\ \tilde{V} \end{pmatrix} \tilde{D} \begin{pmatrix} I \\ \tilde{V} \end{pmatrix}^t + \begin{cases} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ \tilde{v}_1 & \tilde{w} \end{pmatrix} B \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ \tilde{v}_1 & \tilde{w} \end{pmatrix}^t, \\ \text{or} \\ \sigma' \begin{pmatrix} 0 \\ \tilde{w} \end{pmatrix} (0, \tilde{w}^t) . \end{cases}$$

We take $\tilde{D}_1 = \tilde{D}$ and $\tilde{M}_1 = \begin{pmatrix} I \\ \tilde{v} \end{pmatrix}$. Case (ii) of (2.14) is similar. This process may be continued until the full updated factorization has been attained.

To see that only $O(n^2)$ operations are needed, observe that a small fixed number of arithmetic operations (bounded by b say) are required to obtain a new diagonal block. Manipulating the columns of the triangular matrix M at step k requires some fixed multiple of (n - k) arithmetic operations (bounded by a say). Thus, there are at most

$$a(n + (n-1) + \ldots + 1) + bn = \frac{an(n+1)}{2} + bn$$

arithmetic operations required. □

We remark now that the implementation does not actually rewrite $C^{(2)}$ as in (2.16). Instead, (2.15) is written as

$$C^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ v_1 & I & w_1 \\ v_2 & V & w_2 \end{pmatrix} \begin{pmatrix} \delta & 0 & \beta \\ 0 & D_2 & 0 \\ \beta & 0 & \sigma \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ v_1 & I & w_1 \\ v_2 & V & w_2 \end{pmatrix}^t \tag{2.17}$$

$$= \left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & I & 0 \\ \hline v_2 & V & w_2 \end{array}\right) \left(\begin{array}{cc|c} \multicolumn{2}{c|}{\multirow{2}{*}{$\bar{D}$}} & b \\ & & \\ \hline \multicolumn{2}{c|}{b^t} & \sigma \end{array}\right) \left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & I & 0 \\ \hline \tilde{v}_2 & V & \tilde{w}_2 \end{array}\right)^t ,$$

where $$\begin{pmatrix} \bar{D} & b \\ b^t & \sigma \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ v_1 & I & w_1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \delta & 0 & \beta \\ 0 & D_2 & 0 \\ \beta & 0 & \sigma \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ v_1 & I & w_1 \\ 0 & 0 & 1 \end{pmatrix}^t .$$

Multiplying the matrix factors and then equating matrix elements will show that $\bar{D}$ in (2.17) is equal to the matrix $\bar{D}$ appearing in expression (2.11) of Lemma (2.3) if we had first obtained (2.16) and then applied Lemma (2.3). After this form of $C^{(2)}$ has been obtained, the factorization may proceed as described in Lemma (2.3).

We are ready now to give an Algol-like description of the implemented algorithm. Some of the details have been left out for the sake of simplicity. The most notable of these omissions is that when updating a diagonal block D_k we may obtain two 1×1 blocks instead of a 2×2. The explicit bookkeeping involved is not present in this somewhat simplified description.

In the following description of the algorithm we shall make the following conventions:

(1) The expression a : = b means b overwrites a.

(2) D will be a matrix of order at most 4. An expression of the form $\hat{D} : = \begin{pmatrix} \hat{D} & 0 \\ 0 & \sigma \end{pmatrix}$ will mean we have increased the size of $\hat{D}$ with elements defined as indicated. Similar remarks will apply to the arrays V and B.

(3) At step k, w will always have the form $w = \begin{pmatrix} 0 \\ w_1 \\ w_2 \end{pmatrix}$, where w_1 has 1 or 2 components whenever D_k is 1×1 or 2×2 respectively. Matrices Q and Q_k are permutation matrices.

Let $QAQ^t = \sum_{j=1}^{r} M_j D_j M_j^t$, σ, z be given. Let $0 < \alpha < 1$ be fixed. The following algorithm will compute $\tilde{M}_j \tilde{D}_j \tilde{M}_j^t$ and $\tilde{Q}$ such that $\tilde{Q}(A + \sigma zz^t)\tilde{Q}^t = \sum_{j=1}^{\tilde{r}} \tilde{M}_j \tilde{D}_j \tilde{M}_j^t$.

(1) begin

$w : = Qz$; $k : = 1$; $j : = 1$;

$$\hat{D} : = \begin{pmatrix} D_1 + \sigma w_1 w_1^t & \sigma w_1 \\ \sigma w_1^t & \sigma \end{pmatrix}; \quad V : = M_1; \quad w : = w - M_1 w_1;$$

(2) L1 : comment decompose $\hat{D}$ as described in Lemma (2.3);

$$\begin{pmatrix} Q_k & 0 \\ 0 & 1 \end{pmatrix} \hat{D} \begin{pmatrix} Q_k^t & 0 \\ 0 & 1 \end{pmatrix} = \bar{M} \begin{pmatrix} \bar{D} & 0 \\ 0 & B \end{pmatrix} \bar{M}^t;$$

$\hat{D} : = \bar{D}$;

(3) if B is 1×1 then

begin

$$(\tilde{M}_k, w) : = \begin{pmatrix} Q_k & 0 \\ 0 & I \end{pmatrix} (V, w) \begin{pmatrix} Q_k^t & 0 \\ 0 & 1 \end{pmatrix} \bar{M};$$

$\sigma : = B$; $\tilde{D}_k : = \hat{D}$;

ℓ : = order of $\tilde{D}_{kj}$; k : = k + 1; j : = j + ℓ;

$\hat{D}$: = $D_k + \sigma w_1 w_1^t$

if (j = n and $\hat{D}$ is 1×1) or (j = n - 1 and

$|\hat{D}_{21}|\alpha > \max(|\hat{D}_{11}|, |\hat{D}_{22}|)$) then go to QUIT;

$$\hat{D} := \begin{pmatrix} D & \sigma w_1 \\ \sigma w_1^t & \sigma \end{pmatrix}; \quad w := w - M_k w_1; \quad V := M_k;$$

Update Q with Q_k;

go to L1;

end;

(4) if B is 2×2 then

begin

$$(M_k, V, w) := \begin{pmatrix} Q_k & 0 \\ 0 & I \end{pmatrix}(V, w)\begin{pmatrix} Q_k & 0 \\ 0 & 1 \end{pmatrix}\overline{M};$$

$\tilde{D}_k$: = $\hat{D}$; ℓ : = size($\tilde{D}_k$); k : = k + 1; j : = j + ℓ;

if j ≥ n then begin $\tilde{D}_k$: = B_{11}; go to QUIT; end;

$$\hat{D} := \begin{pmatrix} B_{11} & 0 & B_{21} \\ 0 & D_k & 0 \\ B_{21} & 0 & B_{22} \end{pmatrix};$$

$$(V, w) := (V, M_k \mid w)\begin{pmatrix} L^{-1} & -w_1 \\ 0 & 1 \end{pmatrix}; \text{ comment where } (V, M_k) = \begin{pmatrix} L \\ M' \end{pmatrix};$$

$$\hat{D} := \begin{pmatrix} L & w_1 \\ 0 & 1 \end{pmatrix}\hat{D}\begin{pmatrix} L^t & 0 \\ w_1^t & 1 \end{pmatrix};$$

go to L1;

end

QUIT:

end.

We refer the reader now to the brief flow diagram (A1) describing the pivoting strategy and to the program listing (A2) in the appendix. The operation count that follows refers to that particular implementation. The results of the operation count are given in Table 1.

By an ℓ-step reduction we shall mean that ℓ columns of $\tilde{M}$ and the corresponding diagonal blocks have been completely determined by the algorithm we have just described. Operations at step k which are carried out on columns of M or the vector w contribute to the $O(n^2)$ portion of the operation count and will be referred to as operations of type-A. Operations needed to update a diagonal block will be called operations of type-B and they contribute only to the linear term in our operation count. We shall consider an operation as a multiplication and an addition; with this convention we are ignoring the important contribution of interchanges to the $O(n^2)$ term. The paths cited refer to the flow diagram (A1) in the appendix.

Table 1 needs some explanation. The counts given under the heading "path 1" refer to a successful one-step reduction without entering paths 2 or 3 (see A1). The operation counts given for paths 2 and 3 include those cases which begin with path 1 and end in paths 2 or 3.

Table 1
Operations Required at Step k

	Path 1	Path 2	Path 3
type-A	$2(n-k)$	$4(n-(k+1)) \leq m \leq 6(n-(k+1))+1$	$10(n-(k+2))+2 \leq m \leq 11(n-(k+2))+2$
type-B	5	$15 \leq m \leq 19$	42
comparisons	1	4	6
reduction	1	2	3

m denotes number of operations.

The best possible situation occurs when Path 1 is taken at each step; we then have that the total operations required are

$$S = \sum_{j=1}^{n} 2(n-j) + 5(n-1)$$

$$= 2\,\frac{n(n-1)}{2} + 5n$$

$$= n^2 + 4n \ .$$

The worst possible case will now be considered.

Suppose

path 1 is taken for $k = j_1,\ldots,j_{k_1}$;

path 2 is taken for $k = \ell_1,\ldots,\ell_{k_2}$;

path 3 is taken for $k = m_1,\ldots,m_{k_3}$;

where $n = k_1 + 2k_2 + 3k_3$.

The total number of operations contributing to the n^2 term is then bounded by the sum S, where

$$\begin{aligned} S \equiv\ & 2(n - j_1 + n - j_2 +\ldots+ n - j_{k_1}) \\ & + 6(n - (\ell_1+1) + n - (\ell_2+1) +\ldots+ n - (\ell_{k_2}+1)) + k_2 \\ & + 11(n - (m_1+2) + n - (m_2+2) +\ldots+ n - (m_{k_3}+2)) + 2k_3 \\ =\ & 3(n - j_1 + n - j_2 +\ldots+ n - j_{k_1}) \\ & + 3(\{n - \ell_1 + n - (\ell_1+1)\} +\ldots+ \{n - \ell_{k_2} + n - (\ell_{k_2}+1)\}) \\ & + 3(\{n - m_1 + n - (m_1+1) + n - (m_1+2)\} +\ldots \\ & \qquad + \{n - m_{k_3} + n - (m_{k_3}+1) + n - (m_{k_3}+2)\}) - 2k_2 - 7k_3 \\ & + 2(n - (m_1+2) +\ldots+ n - (m_{k_3}+2)) \\ & - (n - j_1 +\ldots+ n - j_{k_1}) \ . \end{aligned}$$

Thus

$$S = \frac{3}{2} n^2 - \frac{3}{2} n + 2k_3 n - 2 \sum_{i=1}^{k_3} m_i + \sum_{i=1}^{k_1} j_i - k_1 n - 2k_2 - 11k_3 .$$

Now,

$$2k_3 n - 2 \sum_{i=1}^{k_3} m_i \leq 2k_3 n - 2 \sum_{i=1}^{k_3} (1 + 3(i-1)) = k_3(2n-3k_3) + k_3 ,$$

and

$$\sum_{i=1}^{k_1} j_i - k_1 n \leq \sum_{j=n-k_1+1}^{n} j - k_1 n = -\frac{1}{2} k_1^2 + \frac{1}{2} k_1 .$$

Thus

$$\begin{aligned} S &\leq \frac{3}{2} n^2 - \frac{3}{2} n + k_3(2n-3k_3-10) - \frac{1}{2} k_1^2 + \frac{1}{2} k_1 - 2k_2 \\ &\leq \frac{3}{2} n^2 - \frac{3}{2} n + \frac{1}{3}(n-5)^2 \\ &= \frac{11}{6} n^2 - \frac{29}{6} n + \frac{25}{3} , \end{aligned}$$

where we have maximized the expression $k_3(2n-3k_3-10)$ over $\{k_3 : k_3 \geq 0\}$. The analysis is not valid unless $n \geq 5$.

Let us divide the counts for type-B operations by the corresponding reduction at step k. An upper bound for this number in the worst case is 14. Thus 14n is an upper bound for the number of type-B operations needed. Therefore, the worst case operation count is bounded by

$$\begin{aligned} &\frac{11}{6} n^2 - \frac{29}{6} n + \frac{25}{3} + 14n \\ &= \frac{11}{6} n^2 + \frac{55}{6} n + \frac{25}{3} . \end{aligned}$$

The maximum number of comparisons needed is 2n.

3. A Pictorial Description of the Algorithm

In the last section we gave a formal description and proof of correctness of an algorithm to update the factorization of a symmetric matrix. The main difficulty in obtaining this algorithm was updating the pivoting strategy while maintaining the triangular structure of M and $\tilde{M}$.

The following diagram represents the algorithm at step k.

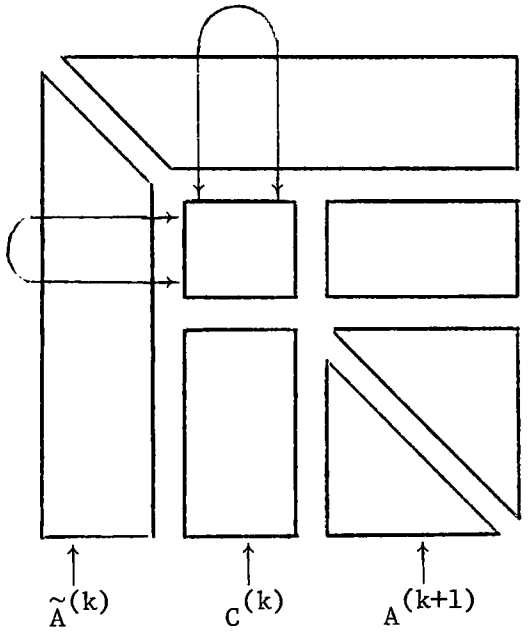
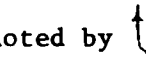


Figure 1
Pivoting in the Updating Algorithm

In Fig. 1, $\tilde{A}^{(k)}$ represents that portion of the factorization of $\tilde{A}$ obtained up to step k. $C^{(k)}$ represents a working array that involves information from the vector w and at most three columns of M. $A^{(k+1)}$ is that portion of the factorization of A which has not yet been considered. From this diagram we see that the pivoting effects neither the triangular structure of $\tilde{M}$ that has already been computed nor the triangular structure of that portion of M which has not yet been

considered.

One can also represent the operations on the elements in a diagram. In the following, d's will represent elements of the diagonal blocks D_k of D, m's will represent elements of the M_j's which occur below the block diagonal of identity matrices in the matrix M, and w's will represent elements of the vector $w = Qz$. Let σ and $A^{(k)}$ be as in Section 2, and we assume that $A^{(k)}$ is in factored form, so only the lower triangle and diagonal D need be stored. A "~" over an element means that some operation has altered this element. If a 0 appears then that element has been "zeroed out" and is not subject to further alteration.

Only those elements that need to be stored are represented; the elements known by definition are left blank; we store the diagonal matrix D in place of the identity matrices in M. Row permutations are denoted by  or  ; column permutations are denoted by  . The permutation matrices Q are not explicitly represented.

We shall illustrate the algorithm with a 5×5 example.

Case 1:

D_1 is 1×1,

(1)
$$\tilde{A} = \left(\begin{array}{c|c} \tilde{d} & \\ \hline m & \\ m & A^{(2)} \\ m & \\ m & \end{array}\right) + \sigma \left(\begin{array}{c} 0 \\ w \\ w \\ w \\ w \end{array}\right) [\qquad\qquad].$$

$\tilde{D}_1$ is computed and found to satisfy the pivoting criteria,

(2)
$$A = \begin{pmatrix} \tilde{d} & & \\ \tilde{m} & & \\ \tilde{m} & A^{(2)} & \\ \tilde{m} & & \\ \tilde{m} & & \end{pmatrix} + \tilde{\sigma} \begin{pmatrix} 0 \\ \tilde{w} \\ \tilde{w} \\ \tilde{w} \\ \tilde{w} \end{pmatrix} [\qquad\qquad].$$

Case 2:

D_1 is 2×2,

(1)
$$\tilde{A} = \begin{pmatrix} \tilde{d} & & \\ \tilde{d} & \tilde{d} & \\ m & m & A^{(2)} \\ m & m & \\ m & m & \end{pmatrix} + \sigma \begin{pmatrix} 0 \\ 0 \\ w \\ w \\ w \end{pmatrix} [\qquad\qquad].$$

$\tilde{D}_1$ has been computed and does not satisfy the criteria for a 2×2 pivot.

(2) Compute $Q_1\tilde{D}_1Q_1^t = \bar{M}\begin{pmatrix} D_1 & 0 \\ 0 & d \end{pmatrix}\bar{M}^t$, $(\tilde{M}_1, V) = \begin{pmatrix} Q_1 & 0 \\ 0 & I \end{pmatrix} M_1 Q_1^t \bar{M}$

$$\begin{pmatrix} Q_1 & 0 \\ 0 & I \end{pmatrix}\tilde{A}\begin{pmatrix} Q_1^t & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \tilde{d} & & \\ \tilde{m} & \tilde{d} & \\ \tilde{m} & m & A^{(2)} \\ \tilde{m} & m & \\ \tilde{m} & m & \end{pmatrix} + \tilde{\sigma} \begin{pmatrix} 0 \\ 0 \\ \tilde{w} \\ \tilde{w} \\ \tilde{w} \end{pmatrix} [\qquad\qquad].$$

We are finished with $\tilde{D}_1$ and are ready to apply the algorithm to the diagonal element in the second position.

Case 3:

$\tilde{D}_1$ is 1×1 and does not satisfy the pivoting criteria with $\tilde{D}_2$ also 1×1.

(1)
$$\tilde{A} = \begin{pmatrix} \tilde{d} & & & \\ m & d & & \\ m & m & & \\ m & m & A^{(3)} & \\ m & m & & \end{pmatrix} + \sigma \begin{pmatrix} 0 \\ w \\ w \\ w \\ w \end{pmatrix} [\qquad\qquad].$$

(2) Compute $(M_1, M_2) = (M_1, M_2)\bar{M}$, $\bar{M}\begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}\bar{M}^t$

$$\tilde{A} = \begin{pmatrix} \tilde{d} & & & \\ \tilde{d} & \tilde{d} & & \\ \tilde{m} & m & & \\ \tilde{m} & m & A^{(3)} & \\ \tilde{m} & m & & \end{pmatrix} + \sigma \begin{pmatrix} 0 \\ 0 \\ \tilde{w} \\ \tilde{w} \\ \tilde{w} \end{pmatrix} [\qquad\qquad].$$

(3) Apply Case 2.

Case 4:

$\tilde{D}_1$ is 1×1 and does not satisfy the pivoting criteria, and $\tilde{D}_2$ is 2×2.

(1)
$$\tilde{A} = \begin{pmatrix} \tilde{d} & & & \\ m & d & & \\ m & d & d & \\ m & m & m & A^{(3)} \\ m & m & m & \end{pmatrix} + \sigma \begin{pmatrix} 0 \\ \tilde{w} \\ \tilde{w} \\ \tilde{w} \\ \tilde{w} \end{pmatrix} [\qquad\qquad].$$

(2) Compute $(\hat{M}_1, \hat{M}_2) = (M_1, M_2)L$, $L\begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}L^t$

$$\tilde{A} = \left(\begin{array}{ccc|c} \tilde{d} & & & \\ \tilde{d} & \tilde{d} & & \\ \tilde{d} & \tilde{d} & \tilde{d} & \\ \hline \tilde{m} & m & m & \\ \tilde{m} & m & m & A^{(3)} \end{array}\right) + \sigma \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline \tilde{w} \\ \tilde{w} \end{array}\right) [\qquad].$$

(3) Compute Q_1, $\bar{M}$ such that $Q_1 D Q_1^t = \bar{M}\begin{pmatrix} \tilde{D}_1 & 0 \\ 0 & D_2 \end{pmatrix}\bar{M}^t$

In (3) the diagonal pivoting strategy could have produced several different block structures for $\tilde{D}_1$ and $\tilde{D}_2$ depending on the matrix D. We only show the case $\tilde{D}_1$ is 2×2 and $\tilde{D}_2$ is 1×1.

$$\begin{pmatrix} Q_1 & 0 \\ 0 & I \end{pmatrix}\tilde{A}\begin{pmatrix} Q_1^t & 0 \\ 0 & I \end{pmatrix} = \left(\begin{array}{ccc|c} \tilde{d} & & & \\ \tilde{d} & \tilde{d} & & \\ \hline \tilde{m} & \tilde{m} & \tilde{d} & \\ \tilde{m} & \tilde{m} & \tilde{m} & \\ \tilde{m} & \tilde{m} & \tilde{m} & A^{(3)} \end{array}\right) + \tilde{\sigma} \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline \tilde{w} \\ \tilde{w} \end{array}\right) [\qquad].$$

(4) We are finished with D_1 and are ready to apply the algorithm to the diagonal element in the third position.

Chapter III

Error Analysis of the Updating Algorithm

1. Introduction

We have updated the symmetric indefinite factorization of

$$\tilde{A} = A + \sigma z z^t \tag{1.1}$$

in order to solve the linear system

$$\tilde{A}x = b \ . \tag{1.2}$$

A method for solving (1.2) is considered to be stable if the computed result x_c satisfies

$$(\tilde{A}+E)x_c = b \tag{1.3}$$

where $\|E\|$ is small compared to $\|\tilde{A}\|$. ($\|\cdot\|$ is the matrix norm induced by a vector norm on R^n which we also denote by $\|\cdot\|$.)

The following analysis is influenced by the error analysis of the diagonal pivoting method given by Bunch [3]. The solution to (1.2) is given in four steps:

$$\begin{aligned} &\text{(i)} && \tilde{A} = \tilde{M}\tilde{D}\tilde{M}^t && \text{(update the decomposition)}, \\ &\text{(ii)} && \tilde{M}c = b && \text{(find the new right-hand side c)}, \\ &\text{(iii)} && \tilde{D}y = c && \text{(solve the 1×1 and 2×2 systems)}, \\ &\text{(iv)} && \tilde{M}^t x = y && \text{(obtain the final solution x)}. \end{aligned} \tag{1.4}$$

We have presented an algorithm that is algebraically correct for obtaining (i). There are standard methods for solving (ii), (iii), and (iv). However, in finite precision arithmetic error is introduced at each of the steps (i) - (iv).

Instead of obtaining the exact decomposition of $\tilde{A}$, we actually obtain $\hat{M} = (\tilde{M}+\Delta\tilde{M})$, and $\hat{D} = (\tilde{D}+\Delta\tilde{D})$ such that $\hat{M}\hat{D}\hat{M}^t = \tilde{Q}\tilde{A}\tilde{Q}^t + S$. Then when equations (ii), (iii), and (iv) in (1.4) are solved, the errors $\delta\hat{M}_1$, $\delta\hat{D}$, $\delta\hat{M}_2$ are introduced at steps (ii), (iii), and (iv), respectively.

Thus, we actually compute $\hat{M}$, $\hat{D}$, c, y, x such that

$$
\begin{aligned}
&\text{(i)} && \hat{M}\hat{D}\hat{M}^t = \tilde{Q}(A+\sigma zz^t)\tilde{Q} + S, \\
&\text{(ii)} && (\hat{M}+\delta\hat{M}_1)c = b, \\
&\text{(iii)} && (\hat{D}+\delta\hat{D})y = c, \\
&\text{(iv)} && (\hat{M}+\delta\hat{M}_2)x = y.
\end{aligned} \tag{1.5}
$$

Now, $\tilde{M}$ and $\tilde{D}$ are the exact factors of $\tilde{A}$. Therefore, steps (i) - (iv) give the exact solution to the system $(\tilde{A}+F)x = b$, where

$$
\begin{aligned}
F = {} & (\Delta\tilde{M}+\hat{M}_1)[\tilde{D} + (\Delta\tilde{D}+\delta\hat{D})][\tilde{M} + (\Delta M+\delta\hat{M}_2)]^t \\
& + \tilde{M}(\Delta\tilde{D}+\delta\hat{D})[\tilde{M} + (\Delta\tilde{M}+\sigma\hat{M}_2)]^t \\
& + \tilde{M}\tilde{D}(\Delta\tilde{M}+\delta\hat{M}_2)^t + S.
\end{aligned} \tag{1.6}
$$

In this chapter if $x = a_1\varepsilon + a_2\varepsilon^2 + \ldots + a_k\varepsilon^k$, where $a_1 \neq 0$, then we write $x = O(\varepsilon)$ and say x is of order ε as $\varepsilon \to 0$. If B is an $n\times n$ matrix with elements b_{ij}, then we shall denote $\tilde{B} = O(\varepsilon)B$ if $\tilde{b}_{ij} = b_{ij}\phi_{ij}(\varepsilon)$, where $\phi_{ij}(\varepsilon) = O(\varepsilon)$. In the following analysis we shall obtain expressions of the form

$$
\begin{aligned}
&\text{(i)} && (\Delta\tilde{M}+\delta\hat{M}_1) = O(\varepsilon)\tilde{M} + G(\varepsilon), \\
&\text{(ii)} && (\Delta\tilde{D}+\delta\hat{D}) = O(\varepsilon)\tilde{D} + H(\varepsilon), \\
&\text{(iii)} && (\Delta\tilde{M}+\delta\hat{M}_2)^t = O(\varepsilon)\tilde{M}^t + G(\varepsilon).
\end{aligned} \tag{1.7}
$$

Using (1.7) in (1.6) gives

$$
\begin{aligned}
(1.8) \qquad F &= (O(\varepsilon)\tilde{M} + G(\varepsilon))[\tilde{D} + O(\varepsilon)D][M + O(\varepsilon)\tilde{M}^t + G(\varepsilon)]^t \\
&\quad + \tilde{M}(O(\varepsilon)D)[\tilde{M} + O(\varepsilon)\tilde{M} + G(\varepsilon)]^t \\
&\quad + \tilde{M}D(O(\varepsilon)\tilde{M} + G(\varepsilon))^t + \hat{M}H(\varepsilon)\hat{M}^t + S \\
&= O(\varepsilon)\tilde{\tilde{M}}\tilde{D}\tilde{M}^t + G(\varepsilon)\tilde{D}\tilde{M}^t \\
&\quad + \tilde{M}(O(\varepsilon)D)\tilde{M}^t \\
&\quad + \tilde{\tilde{M}}\tilde{D}(O(\varepsilon)M^t) + \tilde{\tilde{M}}\tilde{D}[G(\varepsilon)]^t + \tilde{M}H(\varepsilon)\tilde{M}^t \\
&\quad + O(\varepsilon^2)B + S.
\end{aligned}
$$

The $O(\varepsilon^2)B$ term in (1.8) is negligible when compared to the dominant first order terms. The combined terms give

$$F = O(\varepsilon)\tilde{\tilde{M}}\tilde{D}\tilde{M}^t + O(\varepsilon^2)B + S,$$

if $G(\varepsilon) = O(\varepsilon)\tilde{M}$, $H(\varepsilon) = O(\varepsilon)\tilde{D}$, and $S = O(\varepsilon)\tilde{A}$. Then

$$(1.9) \qquad \frac{\|E\|}{\|\tilde{A}\|} = O(\varepsilon);$$

hence the method is stable.

However, we shall also see that the terms S, G, and E will involve products of the entries of solutions to triangular systems involving the original factor M. Thus if M is ill-conditioned, ($\|M\|\,\|M^{-1}\|$ is large compared to the number of significant digits available in our finite precision arithmetic) then the updating procedure cannot guarantee that the constant in the $O(\varepsilon)$ term in (1.9) is of moderate size.

2. A Detailed Description of the Updating Algorithm

We shall now give a detailed floating point analysis of the computations performed in our updating algorithm. There are two parts to a step of the algorithm. An intermediate step of the algorithm

results in a sum of matrices of the form

$$\tilde{Q}_k \tilde{\tilde{A}} \tilde{\tilde{Q}}_k^t = \tilde{A}^{(k-1)} + C_k + A^{(k+\ell)} , \tag{2.1}$$

with

$$\tilde{A}^{(k-1)} = \sum_{j=1}^{k-1} \tilde{M}_j \tilde{D}_j \tilde{M}_j^t, \quad A^{(k+\ell)} = \sum_{j=k+\ell}^{m} M_j D_j M_j^t ,$$

where ℓ is 1 or 2. Let $w^{(j)} = \begin{pmatrix} w_1^{(j)} \\ w_2^{(j)} \end{pmatrix}$ for $1 \leq j \leq m$, where $w = \sum_{j=1}^{m} M_j w_1^{(j)}$ and $\begin{pmatrix} 0 \\ w^{(j+1)} \end{pmatrix} = \begin{pmatrix} 0 \\ w^{(j)} \end{pmatrix} - M_j w_1^{(j)}$.

Part 1 of a step consists in preparing the matrix C_k for part 2. This involves possibly bringing the term $M_{k+\ell} D_{k+\ell} M_{k+\ell}^t$ into the matrix C_k and performing certain operations on the factors of C_k to obtain a special form. Part 2 consists in permuting certain columns and elements of the factors of C_k and obtaining the updated $\tilde{M}_k$ and $\tilde{D}_k$.

We shall now describe an intermediate step in detail.

Part 1.

The previous steps of the algorithm have resulted in

$$\text{(a)} \quad C_k = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v^{(k)} & w^{(k+1)} \end{pmatrix} \begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v^{(k)} & w^{(k+1)} \end{pmatrix}^t$$

with $|\delta| < \alpha|\beta|$, or in

$$\text{(b)} \quad C_k = \begin{pmatrix} 0 \\ w^{(k)} \end{pmatrix} [\sigma] [0, \; w^{(k)t}] .$$

We shall now drop the superscripts. If (a) holds then we replace C_k by

$$C_k = \left(\begin{array}{c|c|c} 0 & & 0 \\ \hline 1 & M_{k+1} & 0 \\ \hline v & & w \end{array}\right)\left(\begin{array}{ccc} \delta & 0 & \beta \\ 0 & D_{k+1} & 0 \\ \beta & 0 & \sigma \end{array}\right)\left(\begin{array}{c|c|c} 0 & & 0 \\ \hline 1 & M_{k+1} & 0 \\ \hline v & & w \end{array}\right)^t$$

$$= \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ v_1 & I & w_1 \\ v_2 & M' & w_2 \end{pmatrix}\begin{pmatrix} \delta & 0 & \beta \\ 0 & D_{k+1} & 0 \\ \beta & 0 & \sigma \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ v_1 & I & w_1 \\ v_2 & M' & w_2 \end{pmatrix}^t$$

where we have partitioned

$$v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \quad w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \quad M_{k+1} = \begin{pmatrix} 0 \\ 0 \\ I \\ M' \end{pmatrix}.$$

We then compute

$$C_k = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & I & 0 \\ v_2-M'v_1 & M' & w_2-M'w_1 \end{pmatrix}\hat{(D)}\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & I & 0 \\ v_2-M'v_1 & M' & w_2-M'w_1 \end{pmatrix}^t, \tag{2.2}$$

where

$$\hat{(D)} = \begin{pmatrix} \delta & \delta v_1^t + \beta w_1^t & \beta \\ v_1\delta + w_1\beta & D_{k+1} + \delta v_1 v_1^t + \beta(v_1 w_1^t + w_1 v_1^t) + \sigma w_1 w_1^t & v_1\beta + w_1\sigma \\ \beta & \beta v_1^t + \sigma w_1^t & \sigma \end{pmatrix}.$$

Now we proceed to part 2.

If (b) holds then we replace C_k by

$$C_k = \left(\begin{array}{c|c} M_k & \begin{matrix} 0 \\ \\ w \end{matrix} \end{array}\right)\begin{pmatrix} D_k & 0 \\ 0 & \sigma \end{pmatrix}\left(\begin{array}{c|c} M_k & \begin{matrix} 0 \\ \\ w \end{matrix} \end{array}\right)^t$$

$$\begin{pmatrix} 0 & 0 \\ I & w_1 \\ M' & w_2 \end{pmatrix}\begin{pmatrix} D_k & 0 \\ 0 & \sigma \end{pmatrix}\begin{pmatrix} 0 & 0 \\ I & w_1 \\ M' & w_2 \end{pmatrix}^t,$$

where we have partitioned

$$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \quad M_k = \begin{pmatrix} 0 \\ I \\ M' \end{pmatrix}.$$

We then compute

$$C_k = \begin{pmatrix} 0 & 0 \\ I & 0 \\ M' & w_2 - M'w_1 \end{pmatrix}\begin{pmatrix} D_k + \sigma w_1 w_1^t & \sigma w_1 \\ \sigma w_1^t & \sigma \end{pmatrix}\begin{pmatrix} 0 & 0 \\ I & 0 \\ M' & w_2 - M'w_1 \end{pmatrix}^t \tag{2.3}$$

and proceed to part 2.

Part 2.

Part 1 has resulted in a matrix of the form

$$C_k = \begin{pmatrix} 0 & 0 \\ I & 0 \\ V & w \end{pmatrix}\begin{pmatrix} \overline{D} & b \\ b^t & \sigma \end{pmatrix}\begin{pmatrix} 0 & 0 \\ I & 0 \\ V & w \end{pmatrix}^t, \tag{2.4}$$

where $\overline{D}$ is a symmetric matrix of order at most 3. We then apply pivoting strategy S_α only to the matrix $\overline{D}$ producing a permutation matrix Q, and a 1×1 or a 2×2 matrix $\tilde{D}$ such that

$$\overline{C}_k = \begin{pmatrix} I & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & I \end{pmatrix} C_k \begin{pmatrix} I & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & I \end{pmatrix}^t$$

is given by one of the following forms:

(a) $\overline{D} = (\delta)$ is 1×1, $b = (\beta)$, and $|\delta| \geq \alpha|\beta|$.

$$\overline{C}_k = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v+w(\beta/\delta) & w \end{pmatrix} \begin{pmatrix} \delta & 0 \\ 0 & \tilde{\sigma} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v+w(\beta/\delta) & w \end{pmatrix}^t \tag{2.5}$$

with $\tilde{\sigma} = \sigma-\beta^2/\delta$. We then take $\tilde{D}_k = (\delta)$, $\tilde{M}_k = \begin{pmatrix} 0 \\ 1 \\ v+w(\beta/\delta) \end{pmatrix}$, and σ is replaced with $\tilde{\sigma}$. Return to part 1.

(b) $Q\overline{D}Q^t = \begin{pmatrix} \delta_{11} & \delta_{21} \\ \delta_{21} & \delta_{22} \end{pmatrix}$, $b^t = (\beta_1,\beta_2)$, $|\delta_{11}| \geq |\delta_{22}|$, and $|\delta_{11}| \geq \alpha|\delta_{21}|$. Then

$$\overline{C}_k = L_k \begin{pmatrix} \delta_{11} & 0 \\ 0 & B \end{pmatrix} L_k^t, \tag{2.6}$$

where

$$B = \begin{pmatrix} \delta_{22} - \delta_{21}^2/\delta_{11} & \beta_2 - (\beta_1\delta_{21})/\delta_{11} \\ \beta_2 - (\beta_1\delta_{21})/\delta_{11} & \sigma - \beta_1^2/\delta_{11} \end{pmatrix},$$

$$L_k = \left(\begin{array}{ccc|c|c} & 0 & & 0 & 0 \\ & 1 & & 0 & 0 \\ & \delta_{21}/\delta_{11} & & 1 & 0 \\ \hline v_1 + v_2(\delta_{21}/\delta_{11}) + w(\beta_1/\delta_{11}) & & & v_2 & w \end{array}\right),$$

and we have partitioned $V = (v_1,v_2)$. We then take

$$\tilde{D}_k = (\delta_{11}),\ \tilde{M}_k = \begin{pmatrix} 0 \\ 1 \\ \delta_{21}/\delta_{11} \\ v_1 + v_2(\delta_{21}/\delta_{11}) + w(\beta_1/\delta_{11}) \end{pmatrix}, \text{ and}$$

$$C_{k+1} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ v_2 & w \end{pmatrix} (B) \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ v_2 & w \end{pmatrix}^t .$$

Return to part 1.

(c) $\bar{D} = \begin{pmatrix} \delta_{11} & \delta_{21} \\ \delta_{21} & \delta_{22} \end{pmatrix}$ and $\max(|\delta_{11}|, |\delta_{22}|) < \alpha|\delta_{21}|$.

$$\bar{C}_k = \begin{pmatrix} 0 & 0 \\ I & 0 \\ V + wb^t\bar{D}^{-1} & w \end{pmatrix} \begin{pmatrix} \bar{D} & 0 \\ 0 & \sigma - b^t\bar{D}^{-1}b \end{pmatrix} \begin{pmatrix} 0 & 0 \\ I & 0 \\ V + wb^t\bar{D}^{-1} & w \end{pmatrix}^t . \tag{2.7}$$

We then take

$$\tilde{D}_k = \bar{D},\ \tilde{M}_k = \begin{pmatrix} 0 \\ I \\ V + wb^t\bar{D}^{-1} \end{pmatrix},$$

and replace σ by $\sigma - b^t\bar{D}^{-1}b$.

(d) $Q\bar{D}Q^t = \begin{pmatrix} \delta_{11} & \delta_{21} & \delta_{31} \\ \delta_{21} & & \\ \delta_{31} & & \bar{D}_2 \end{pmatrix}$, δ_{11} is the pivot choice when S_α is applied to $\bar{D}$, $|\delta_{11}| \geq \alpha|\delta_{i1}|$ for i=2,3; and $b^t = (\beta_1, b_2^t)$.

$$(2.8)\qquad \overline{C}_k = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ \delta_{21}/\delta_{11} & 1 & 0 & 0 \\ \delta_{31}/\delta_{11} & 0 & 1 & 0 \\ \tilde{v}_1 & v_2 & v_3 & w \end{pmatrix} \begin{pmatrix} \delta_{11} & 0 & 0 \\ 0 & \hat{D}_2 & \hat{b}_2 \\ 0 & \hat{b}_2^t & \tilde{\sigma} \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ \delta_{21}/\delta_{11} & 1 & 0 & 0 \\ \delta_{31}/\delta_{11} & 0 & 1 & 0 \\ \tilde{v}_1 & v_2 & v_3 & w \end{pmatrix}^t$$

where $\tilde{v}_1 = v_1 + v_2(\delta_{21}/\delta_{11}) + v_3(\delta_{31}/\delta_{11}) + w(\beta_1/\delta_{11})$,

$$\hat{D}_2 = \overline{D}_2 - \frac{1}{\delta_{11}} \begin{pmatrix} \delta_{21} \\ \\ \delta_{31} \end{pmatrix} (\delta_{21},\ \delta_{31}) \ ,$$

$$\hat{b}_2^t = b_2^t - (\beta_1/\delta_{11})(\delta_{21},\ \delta_{31}) \ ,$$

$$\tilde{\sigma} = \sigma - \beta_1^2/\delta_{11} \ ,$$

and we have partitioned $V = (v_1, v_2, v_3)$.

We then take

$$\tilde{D}_k = (\delta_{11}),\ \tilde{M}_k = \begin{pmatrix} 0 \\ 1 \\ \delta_{21}/\delta_{11} \\ \delta_{31}/\delta_{11} \\ \tilde{v}_1 \end{pmatrix} ,$$

and

$$C_{k+1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ v_1 & v_2 & w \end{pmatrix} \begin{pmatrix} \tilde{D}_2 & \tilde{b}_2 \\ \tilde{b}_2^t & \tilde{\sigma} \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ v_1 & v_2 & w \end{pmatrix}^t .$$

We now bypass part 1 since C_{k+1} is already in the proper form needed to apply part 2.

(e) $Q\bar{D}Q^t = \begin{pmatrix} \bar{D}_1 & \begin{matrix}\delta_{31}\\ \delta_{32}\end{matrix} \\ \begin{matrix}\delta_{31} & \delta_{32}\end{matrix} & \delta_{33} \end{pmatrix}$; S_α applied to $\bar{D}$ resulted in the choice of a 2×2 pivot D_1; $|\det \bar{D}_1| \geq (1-\alpha^2)(\max|d_{ij}|)^2$, and $b^t = (b_1^t, \beta_2)$.

$$(2.9)\quad \bar{C}_k = \begin{pmatrix} 0 & 0 & 0 \\ I & 0 & 0 \\ \tilde{d}^t & 1 & 0 \\ V_1 + v_2\tilde{d}^t + w\tilde{b}_1^t & v_2 & w \end{pmatrix} \begin{pmatrix} \bar{D}_1 & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ I & 0 & 0 \\ \tilde{d}^t & 1 & 0 \\ V_1 + v_2\tilde{d}^t + w\tilde{b}_1^t & v_2 & w \end{pmatrix}^t$$

where

$$\tilde{d}^t = (\delta_{31}, \delta_{32})\bar{D}_1^{-1},$$

$$\tilde{b}_1^t = b_1^t\bar{D}_1^{-1},$$

$$B = \begin{pmatrix} \delta_{33} - (\delta_{31}, \delta_{32})\bar{D}_1^{-1}\begin{pmatrix}\delta_{31}\\ \delta_{32}\end{pmatrix} & \beta_2 - b_1^t\bar{D}_1^{-1}\begin{pmatrix}\delta_{31}\\ \delta_{32}\end{pmatrix} \\ \beta_2 - b_1^t\bar{D}_1^{-1}\begin{pmatrix}\delta_{31}\\ \delta_{32}\end{pmatrix} & \sigma - b_1^t\bar{D}_1^{-1}b_1 \end{pmatrix},$$

and we have partitioned $V = (V_1, v_2)$.

We then take

$$\tilde{D}_k = \bar{D}_1, \quad \tilde{M}_k = \begin{pmatrix} 0 \\ I \\ \tilde{d}^t \\ V_1 + v_2\tilde{d}^t + w\tilde{b}_1^t \end{pmatrix}$$

and

$$C_{k+1} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ v_2 & w \end{pmatrix} (B) \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ v_2 & w \end{pmatrix}^t .$$

Return to part 1.

3. Floating Point Analysis

Now that we have a detailed description of the numerical operations performed, we are ready to examine the error introduced when these operations are carried out in finite precision floating point arithmetic. We shall work in base β, t-digit floating point arithmetic. We call $\varepsilon \equiv \frac{1}{2}\beta^{1-t}$ the basic machine unit. Let

$$F\ell(\beta,t) = \{\theta: \theta = \pm\beta^k\left(\sum_{j=1}^{t} \gamma_j \beta^{-j}\right), \quad 0 \le \gamma_j \le (\beta-1) ,$$

each δ_j an integer,

$1 \le \gamma_1 \le (\beta-1)$, k any integer$\}$.

We then have [10]

$$f\ell(\theta_1 * \theta_2) = (\theta_1 * \theta_2)(1+\varepsilon') ,$$

where $|\varepsilon'| \le \varepsilon$, whenever $\theta_1, \theta_2 \in F\ell(\beta,t)$ are floating point numbers, and $*$ is one of the operations $\{+, -, \cdot, /\}$, and $f\ell(\theta_1 * \theta_2)$ is the nearest number in $F\ell(\beta,t)$ to the real number $\theta_1 * \theta_2$. We shall also write $f\ell(B)$ to denote the computed elements of the matrix (or vector) B.

Lemma (3.1). Consider the vector $v^{(k)}$ defined in part 1(a) of section 2. Then the components $v_i^{(k)}$ of $v^{(k)}$ satisfy

$$fl(v_i^{(k)}) = v_i^{(k)} + \tau_i^{(k)}(\varepsilon) ,$$

where

$$|\tau_i^{(k)}(\varepsilon)| \leq (3+\varepsilon)\varepsilon(i-1)\max_{l\leq k}|v_i^{(l)}| . \tag{3.1}$$

Proof. The vector $v^{(k)}$ is the vector v_2 appearing in one and only one of the expressions (2.6) or (2.9) at step k-1. The vector v_2 in (2.6) or (2.9) is one of the columns of the matrix V in the expression (2.4). Since V in (2.4) is given by (2.2) or by (2.3), we see that

$$V = (v_2^{(k-1)} - M'v_1^{(k-1)}) \quad \text{or} \quad V = M', \tag{3.2}$$

where

$$\begin{pmatrix} 0 \\ I \\ M' \end{pmatrix} = M_{k+1}, \quad \text{and} \quad \begin{pmatrix} v_1^{(k-1)} \\ v_2^{(k-1)} \end{pmatrix} = v^{(k-1)} .$$

If $v^{(k)}$ is defined as a column of M, then no error is introduced. However, if

$$v^{(k)} = v_2^{(k-1)} - M'v_1^{(k-1)}, \tag{3.3}$$

then let j be the largest index less than k for which $v^{(j)}$ was defined by a column of M. Then

$$\begin{pmatrix} 0 \\ v^{(k)} \end{pmatrix} = \begin{pmatrix} 0 \\ v^{(j)} \end{pmatrix} - \sum_{l=j+1}^{k} M_l v_1^{(l-1)} , \tag{3.4}$$

where we have partitioned $v^{(l)} = \begin{pmatrix} v_1^{(l)} \\ v_2^{(l)} \end{pmatrix}$ so that $M_l v_1^{(l-1)}$ makes sense. The formula (3.4) can be derived from (3.3) using an inductive argument. We recognize (3.4) as the process we would use to solve the linear system

$$(3.5) \qquad Mx = \begin{bmatrix} 0 \\ v^{(j)} \end{bmatrix} ,$$

where $v^{(k)}$ is the result of the k^{th} step of that process. Then it has been shown [6] that

$$f\ell(v_i^{(k)}) = v_i^{(k)} + \tau_i^{(k)}(\varepsilon) ,$$

where

$$(3.6) \qquad |\tau_i^{(k)}(\varepsilon)| \leq (3+\varepsilon)\varepsilon(i-1) \max_{j \leq \ell \leq k} |v_i^{(\ell)}| .$$

The bound (3.1) follows from (3.6), but is not as good as (3.6); however, (3.6) cannot be obtained without prior knowledge of the index j. □

We shall comment now on the growth of the $v_i^{(\ell)}$ in (3.4). Let us consider equation (3.5) further. Since $v^{(j)}$ is defined by that portion of some column of M (say column i) which lies below the main diagonal, we may write (3.5) as

$$Mx = Me_i - e_i ,$$

where e_i is the basis vector defined by $(e_i)_j = \begin{cases} 0 \text{ if } j \neq i \\ 1 \text{ if } j = i \end{cases}$. Thus

$$M(x-e_i) = -e_i .$$

Therefore, the solution $e_i - x$ is a column of M^{-1}. This shows that the $v_1^{(\ell)}$ in (3.4) are in fact composed of elements of M^{-1}. We now observe that undue growth in the $v_1^{(\ell)}$ in (3.4) indicates severe ill-conditioning of the matrix M with respect to solving linear equations.

Lemma (3.2). Consider the computed quantities $f\ell(w_1^{(k)})$. Then if

$$\tilde{p} \equiv \begin{bmatrix} f\ell(w^{(1)}) \\ \vdots \\ f\ell(w_1^{(m)}) \end{bmatrix} ,$$

we have the $\tilde{p}$ satisfies the equation

$$(M+T)\tilde{p} = w \ ,$$

where the elements τ_{ij} of T satisfy

$$|\tau_{ij}| \leq (n+1)\gamma|m_{ij}|\varepsilon$$

Here, γ is a constant of order unity and the m_{ij} are the elements of M.

Proof: One observes that the $w_1^{(j)}$ are computed from the standard back substitution algorithm. The result then follows from [20]. □

Let us drop the $f\ell$-notation and hereafter regard the quantities $w_1^{(j)}$ as computed quantities. Then Lemma (3.2) shows that we may write

$$\begin{aligned} MDM^t + \sigma ww^t = M\Big(D + \sigma \begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_1^{(m)} \end{bmatrix} [w_1^{(1)} \ldots w_1^{(m)}]\Big)M^t \\ + \sigma\Big(T \begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_1^{(m)} \end{bmatrix} w^t + w[w_1^{(1)} \ldots w_1^{(m)}]T^t\Big) \\ + \sigma T \begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_1^{(m)} \end{bmatrix} [w_1^{(1)} \ldots w_1^{(m)}]T^t \ . \end{aligned} \tag{3.7}$$

Thus we shall now regard the vectors $w^{(k)}$ as exact quantities. The error introduced from the computation of these quantities in finite precision is expressed in the error matrix

$$S = \sigma\{T\tilde{p}w^t + w\tilde{p}^tT^t + T\tilde{\tilde{p}}\tilde{p}^tT^t\} \ .$$

Lemma (3.3). Let $v_i^{(k)}$ denote a component of $v_1^{(k)}$. (See Lemma (3.1), equation (3.2).) Let $\omega_j^{(k)}$ denote a component of $w_1^{(k)}$. Then the floating point computation of $\tilde{D}$ results in

$$fl(\tilde{D}) = \tilde{D} + E \ ,$$

where E is a block diagonal matrix with the same structure as that of $\tilde{D}$. Moreover, the 2×2 blocks of $fl(\tilde{D})$ satisfy

$$\alpha|\tilde{\delta}_{21}^{(k)}| > \max\left(|\tilde{\delta}_{11}^{(k)}|, |\tilde{\delta}_{22}^{(k)}|\right), \text{ where } \tilde{D}_k = \begin{pmatrix} \tilde{\delta}_{11}^{(k)} & \tilde{\delta}_{21}^{(k)} \\ \tilde{\delta}_{21}^{(k)} & \tilde{\delta}_{22}^{(k)} \end{pmatrix}$$

and the elements ε_{ij} of E satisfy

$$|\varepsilon_{ij}| \leq C\epsilon \ , \tag{3.8}$$

where

$$0 \leq C < \max_{ijk}\left(|\delta_{ij}|, \alpha|\beta_k(v_i^{(k)})^2|, 2|\beta_k v_i^{(k)} w_j^{(k)}|, |\sigma_k(w_i^{(k)})^2|\right)49 \ .$$

(Here σ_k is "σ" at the k^{th} step, and β_k is the "β" appearing in part 1 at the k^{th} step.)

Proof: From equations (2.2) and (2.3) we see that the updated diagonal blocks $\tilde{D}_k$ are obtained by decomposing matrices of the form

$$\hat{D}_1 = \begin{pmatrix} \delta & \delta v_1^t + \beta w_1^t & \beta \\ v_1\delta + w_1\beta & D_{k+1} + \delta v_1 v_1^t + \beta(v_1 w_1^t + w_1 v_1^t) + \sigma w_1 w_1^t & v_1\beta + \sigma w_1 \\ \beta & \beta v_1^t + \sigma w_1^t & \sigma \end{pmatrix} ,$$

with $|\delta| < \alpha|\beta|$ if equation (2.2) was used, or

$$\hat{D}_2 = \begin{pmatrix} D_k + \sigma w_1 w_1^t & \sigma w_1 \\ \sigma w_1^t & \delta \end{pmatrix}$$

if equation (2.3) was used. Here the v's, w's, β, δ, α are the previously computed quantities at step k; we have left off the superscripts.

Then

$$f\ell(\hat{D}_\ell) = \hat{D}_\ell + E_\ell, \ (\ell \text{ is 1 or 2}) \ ,$$

where a typical (i,j) element of $f_\ell(D_1)$ is of the form

$$\Big[\{[\delta_{ij} + \delta\nu_i\nu_j(1+\varepsilon_1)(1+\varepsilon_2)](1+\varepsilon_3) + 2\beta\nu_i\omega_j(1+\varepsilon_4)(1+\varepsilon_5)$$
$$\}(1+\varepsilon_6) + \sigma\omega_i\omega_j(1+\varepsilon_7)(1+\varepsilon_8)\Big](1+\varepsilon_9)$$
$$= \delta_{ij}(1+\varepsilon_3)(1+\varepsilon_6)(1+\varepsilon_9) + \delta\nu_i\nu_j(1+\varepsilon_1)(1+\varepsilon_2)(1+\varepsilon_3)(1+\varepsilon_6)(1+\varepsilon_9)$$
$$+ 2\beta\nu_i\omega_j(1+\varepsilon_4)(1+\varepsilon_5)(1+\varepsilon_6)(1+\varepsilon_9) + \sigma\omega_i\omega_j(1+\varepsilon_7)(1+\varepsilon_8)(1+\varepsilon_9) \ ,$$

where $|\varepsilon_j| \leq \varepsilon$.

Now, if $r\varepsilon < 0.1$ and $|\rho| < \varepsilon$, then $(1+\rho)^r = 1 + r\rho'$, where $|\rho'| < 1.06\varepsilon$ [19, ex. 4, p. 80], and if $|\rho_1|,\ldots,|\rho_r| \leq \varepsilon$ then

$$(1+\rho_1) \ \ldots \ (1+\rho_r) = 1 + r\rho \ ,$$

where $|\rho| < 1.06\varepsilon$. Thus we see that if the elements of $\hat{D}_\ell$ are denoted by $\delta_{ij}^{(\ell)}$ and the elements of E_ℓ are denoted by $\varepsilon_{ij}^{(\ell)}$ then

$$\delta_{ij}^{(1)} + \varepsilon_{ij}^{(1)} = \delta_{ij}(1+3\rho_1) + \delta\nu_i\nu_j(1+5\rho_2) +$$
$$+ 2\beta\nu_i\omega_j(1+4\rho_3) + \sigma\omega_i\omega_j(1+3\rho_4) \ ,$$

where $|\rho_j| < 1.06\varepsilon$ for $j = 1,2,3,4$, and $|\delta| < \alpha|\beta|$. Hence,

$$|\varepsilon_{ij}^{(1)}| \leq \max(|\delta_{ij}|, \ \alpha|\beta\nu_i\nu_j|, \ 2|\beta\nu_i\omega_j|, \ |\sigma\omega_i\omega_j|)15(1.06)\varepsilon \ .$$

Maximizing the quantities which appear in this expression gives the bound

$$(3.9) \qquad |\varepsilon_{ij}^{(\ell)}| \leq \max_{ijk}(|\delta_{ij}|, \ |\alpha\beta_k[\nu_i^{(k)}]^2|, \ 2|\beta_k\nu_i^{(k)}\omega_j^{(k)}|, \ |\sigma_k(\omega_j^{(k)})^2|)15.9\varepsilon.$$

The case we have examined is clearly the worst case for the type of analysis we have carried out. Thus we take (3.9) as our bound for

the elements of E_ℓ, $(\ell = 1,2)$.

The next step in obtaining $\tilde{D}_k$ is given by decomposing $\hat{D}_1$ or $\hat{D}_2$ according to one of the equations (2.5)-(2.9). Let us refer now to the proof of Lemma (2.3) of Chapter II. Specifically we consider the decomposition given in equation (2.12) of Chapter II. There it was shown that a scale factor μ may be implicitly introduced in the last row and column of $\hat{D}_1$ or $\hat{D}_2$. This factor has no consequence on the final result. However, when $\hat{D}_1$ or $\hat{D}_2$ is suitably scaled in this way then pivoting strategy S_α does not choose any of the elements in the last row or column as pivot elements. We then obtain a computed factorization.

$$\hat{M}\hat{D}\hat{M}^t = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}(\hat{D}_i + E_i)\begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}^t + F_i \quad (i = 1 \text{ or } 2).$$

The analysis given in [3] applied to this (at most) 4×4 case shows that the elements $f_{ij}^{(\ell)}$ of F_ℓ satisfy

$$|f_{ij}^{(\ell)}| \leq \max_{ij}|\delta_{ij}^{(\ell)}|(34\varepsilon). \tag{3.10}$$

Now (3.10) together with (3.9) give the bound in (3.8). □

We have given an analysis of all of the operations in part 1 and of the formation of $\tilde{D}$. We now turn to an analysis of the final formation of the elements of M. We begin with

Lemma (3.4). Let $\tilde{m}_{ij}$ be the ij^{th} element of $\tilde{M}$ with $i > j$. Then

$$f\ell(\tilde{m}_{ij}) = \tilde{m}_{ij} + \tilde{m}_{ij}\mu_{ij}(\varepsilon) + \nu_{ij}(\varepsilon), \tag{3.11}$$

where

$$|\mu_{ij}(\varepsilon)| \leq \varepsilon(3+\varepsilon),$$

and

$$|\nu_{ij}(\varepsilon)| \leq \max_{ik} |\nu_i^{(k)}| (\varepsilon(3 + \max[\frac{7}{\alpha}, \frac{1}{1-\alpha}]) + O(\varepsilon^2)) .$$

Proof: We shall give a detailed analysis of the operations used in forming $\tilde{M}$. These operations are described in part 2 a-e. We shall have to examine each case separately. Cases a-e below refer to the operations performed in part 2 a-e, respectively. The errors resulting from computing the quantities β/δ, $b^t\tilde{D}^{-1}$ have been accounted for in Lemma (3.3). Therefore, we shall assume here that we have these quantities exactly.

Case a: The vector $\tilde{M}_k$ is computed by

$$f\ell(\tilde{M}_k) = \begin{pmatrix} 0 \\ 1 \\ f\ell(v^{(k)} + w^{(k+1)}(\beta/\delta)) \end{pmatrix}$$

Now,

$$\begin{aligned} f\ell(\nu_i^{(k)} + \omega_i^{(k+1)}(\beta/\delta)) &= \\ &= (\nu_i^{(k)} + [\omega_i^{(k)}(\beta/\delta)](1+\varepsilon_1))(1+\varepsilon_2) \\ &= [\nu_i^{(k)} + \omega_i^{(k)}(\beta/\delta)](1+\varepsilon_2) + \omega_i^{(k)}(\beta/\delta)(\varepsilon_1+\varepsilon_1\varepsilon_2) \\ &= \tilde{m}_{ik} + \tilde{m}_{ik\ 2} + \omega_i^{(k)}(\beta/\delta)(\varepsilon_1+\varepsilon_1\varepsilon_2) \\ &= \tilde{m}_{ik} + \tilde{m}_{ik}\varepsilon_2 + (\nu_i^{(k)} + \omega_i^{(k)}(\beta/\delta))(\varepsilon_1+\varepsilon_1\varepsilon_2) \\ &\quad - \nu_i^{(k)}(\varepsilon_1+\varepsilon_1\varepsilon_2) \\ &= \tilde{m}_{ik} + \tilde{m}_{ik}(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2) - \nu_i^{(k)}(\varepsilon_1+\varepsilon_1\varepsilon_2) . \end{aligned}$$

Case b: $\tilde{M}_k$ is computed by

$$f\ell(\tilde{M}_k) = \begin{pmatrix} 0 \\ 1 \\ \delta_{21}/\delta_{11} \\ f\ell(v_1 + v_2(\delta_{21}/\delta_{11}) + w^{(k+1)}(\beta_1/\delta_{11})) \end{pmatrix}$$

Let v_j have components v_{ij}, $j = 1,2$.

Now,

$$\begin{aligned}
f\ell(\tilde{m}_{ik}) &= f\ell(v_{i1} + v_{i2}(\delta_{21}/\delta_{11}) + \omega_i^{(k+1)}(\beta_1/\delta_{11})) \\
&= \{[v_{i1} + v_{i2}(\delta_{21}/\delta_{11})(1+\varepsilon_1)](1+\varepsilon_2) \\
&\quad + \omega_i^{(k+1)}(\beta_1/\delta_{11})(1+\varepsilon_3)\}(1+\varepsilon_4) \\
&= [v_{i1}(1+\varepsilon_2) + v_{i2}(\delta_{21}/\delta_{11})(1+\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2) \\
&\quad + \omega_i^{(k+1)}(\beta_1/\delta_{11})(1+\varepsilon_3)](1+\varepsilon_4) \\
&= \tilde{m}_{ik} + \tilde{m}_{ik}\varepsilon_4 \\
&\quad + (1+\varepsilon_4)[v_{i1}\varepsilon_2 + v_{i2}(\delta_{21}/\delta_{11})(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2) \\
&\qquad + \omega_i^{(k+1)}(\beta_1/\delta_{11})\varepsilon_3] \\
&= \tilde{m}_{ik} + \tilde{m}_{ik}\varepsilon_4 \\
&\quad + (1+\varepsilon_4)[(v_{i1} + v_{i2}(\delta_{21}/\delta_{11}) + \omega_i^{(k+1)}(\beta_1/\delta_{11}))\varepsilon_3 \\
&\qquad + v_{i1}(\varepsilon_2-\varepsilon_3) + v_{i2}(\delta_{21}/\delta_{11})(\varepsilon_1+\varepsilon_2-\varepsilon_3+\varepsilon_1\varepsilon_2)] \\
&= \tilde{m}_{ik} + \tilde{m}_{ik}(\varepsilon_3+2\varepsilon_4+\varepsilon_3\varepsilon_4) + v_{i1}(\varepsilon_2-\varepsilon_3+\varepsilon_4(\varepsilon_2-\varepsilon_3)) \\
&\quad + v_{i2}(\delta_{21}/\delta_{11})(\varepsilon_1+\varepsilon_2-\varepsilon_3+\varepsilon_1\varepsilon_2)(1+\varepsilon_4) \cdot
\end{aligned}$$

Case c: $\tilde{M}_k$ is computed by

$$f\ell(\tilde{M}_k) = \begin{pmatrix} 0 \\ I \\ f\ell[(v_1,v_2) + w^{(k+1)}{}_b{}^{t\tilde{D}_k^{-1}}] \end{pmatrix} .$$

Let $b^{t\tilde{D}_k^{-1}} = (\tilde{\beta}_1,\tilde{\beta}_2)$. Then

$$f\ell(\tilde{M}_{ik}) = (f\ell(v_{i1} + \omega_i^{(k+1)}\tilde{\beta}_1),\ f\ell(v_{i2} + \omega_i^{(k+1)}\tilde{\beta}_2)) .$$

Each of the components falls under the same analysis as case a. We obtain

$$f\ell(\tilde{m}_{ij}) = \tilde{m}_{ij} + \tilde{m}_{ij}(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2) - v_{i1}(\varepsilon_1+\varepsilon_1\varepsilon_2)$$

and

$$f\ell(\tilde{m}_{ij+1}) = \tilde{m}_{ij+1} + \tilde{m}_{ij+1}(\varepsilon_1'+\varepsilon_2'+\varepsilon_1'\varepsilon_2') - v_{i2}(\varepsilon_1'+\varepsilon_1'\varepsilon_2') ,$$

where $\tilde{M}_{ik} = (\tilde{m}_{ij},\ \tilde{m}_{ij+1})$.

Case d: $\tilde{M}_k$ is computed by

$$f\ell(\tilde{M}_k) = \begin{pmatrix} 0 \\ 1 \\ \delta_{21}/\delta_{11} \\ \delta_{31}/\delta_{11} \\ f\ell(v_1 + v_2(\delta_{21}/\delta_{11}) + v_3(\delta_{31}/\delta_{11}) + w^{(k+2)}(\beta_1/\delta_{11})) \end{pmatrix} .$$

So

$$\begin{aligned} f\ell(\tilde{m}_{ik}) &= f\ell(v_{i1} + v_{i2}(\delta_{21}/\delta_{11}) + v_{i3}(\delta_{21}/\delta_{11}) + \omega_i^{(k+2)}(\beta_1/\delta_{11})) \\ &= \Big[\{[v_{i1} + v_{i2}(\delta_{21}/\delta_{11})(1+\varepsilon_1)](1+\varepsilon_2) \\ &\quad + v_{i3}(\delta_{31}/\delta_{11})(1+\varepsilon_3)\}(1+\varepsilon_4) + \omega_i^{(k+2)}(\beta_1/\delta_{11})(1+\varepsilon_5)\Big](1+\varepsilon_6) \end{aligned}$$

$$= \{\nu_{i1} + \nu_{i2}(\delta_{21}/\delta_{11}) + \nu_{i3}(\delta_{31}/\delta_{11}) + \omega_i^{(k+2)}(\beta_1/\delta_{11})$$

$$+ \nu_{i1}(\varepsilon_2+\varepsilon_4+\varepsilon_2\varepsilon_4) + \nu_{i2}(\delta_{21}/\delta_{11})(\varepsilon_1+\varepsilon_2+\varepsilon_4+\varepsilon_1\varepsilon_2$$

$$+\varepsilon_4(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2))$$

$$+ \nu_{i3}(\delta_{31}/\delta_{11})(\varepsilon_3+\varepsilon_4+\varepsilon_3\varepsilon_4) + \omega_i^{(k+2)}(\beta_1/\delta_{11})\varepsilon_5\}(1+\varepsilon_6)$$

$$= \tilde{m}_{ik} + \tilde{m}_{ik}\varepsilon_6 + \nu_{i1}[\varepsilon_2+\varepsilon_4+\varepsilon_6+\varepsilon_2\varepsilon_4+\varepsilon_6(\varepsilon_2+\varepsilon_4+\varepsilon_2\varepsilon_4)]$$

$$+ \nu_{i2}\left(\frac{\delta_{21}}{\delta_{11}}\right)[\varepsilon_1+\varepsilon_2+\varepsilon_4+\varepsilon_6+\varepsilon_1\varepsilon_2+\varepsilon_4(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2)$$

$$+\varepsilon_6(\varepsilon_1+\varepsilon_2+\varepsilon_4+\varepsilon_1\varepsilon_2+\varepsilon_4(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2))]$$

$$+ \nu_{i3}\left(\frac{\delta_{31}}{\delta_{11}}\right)[\varepsilon_3+\varepsilon_4+\varepsilon_6+\varepsilon_3\varepsilon_4+\varepsilon_6(\varepsilon_2+\varepsilon_4+\varepsilon_3\varepsilon_4)]$$

$$+ \omega_i^{(k+2)}\left(\frac{\beta_1}{\delta_{11}}\right)(\varepsilon_5+\varepsilon_6+\varepsilon_5\varepsilon_6)$$

$$= \tilde{m}_{ik} + \tilde{m}_{ik}\varepsilon_6 + (\varepsilon_5+\varepsilon_6+\varepsilon_5\varepsilon_6)[\nu_{i1} + \nu_{i2}\left(\frac{\delta_{21}}{\delta_{11}}\right) + \nu_{i3}\left(\frac{\delta_{31}}{\delta_{11}}\right)$$

$$+ \omega_i^{(k+2)}\left(\frac{\beta_1}{\delta_{11}}\right)]$$

$$+ \nu_{i1}[\varepsilon_2+\varepsilon_4-\varepsilon_5+\varepsilon_2\varepsilon_3-\varepsilon_5\varepsilon_6+O(\varepsilon)^2]$$

$$+ \nu_{i2}\left(\frac{\delta_{21}}{\delta_{11}}\right)[\varepsilon_1+\varepsilon_2+\varepsilon_4-\varepsilon_5+\varepsilon_1\varepsilon_2-\varepsilon_5\varepsilon_6+\varepsilon_4(\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2)+O(\varepsilon^2)]$$

$$+ \nu_{i3}\left(\frac{\delta_{31}}{\delta_{11}}\right)[\varepsilon_3+\varepsilon_4-\varepsilon_5+\varepsilon_3\varepsilon_4-\varepsilon_5\varepsilon_6+O(\varepsilon^2)] \ .$$

Case e: $\tilde{M}_k$ is computed by

$$f\ell(\tilde{M}_k) = \begin{pmatrix} 0 \\ I \\ (\delta_{31},\delta_{32})\tilde{D}_k^{-1} \\ f\ell[(v_1,v_2) + v_3(\delta_{31},\delta_{32})\tilde{D}_k^{-1} + w^{(k+2)}(\beta_1,\beta_2)\tilde{D}_k^{-1}] \end{pmatrix}$$

where we have partitioned $V = (v_1, v_2, v_3)$, $b^t = (\beta_1, \beta_2, \beta_3)$. Thus if we let $(\tilde{\delta}_{31}, \tilde{\delta}_{32}) = (\delta_{31}, \delta_{32})\tilde{D}_k^{-1}$, $(\tilde{\beta}_1, \tilde{\beta}_2) = (\beta_1, \beta_2)\tilde{D}_k^{-1}$ be the quantities computed in (2.9), we have

$$fl(\tilde{M}_{ik}) = (fl(v_{i1} + v_{i3}\tilde{\delta}_{31} + \omega_i^{(k+2)}\tilde{\beta}_1),\ fl(v_{i2} + v_{i3}\tilde{\delta}_{32} + \omega_i^{(k+2)}\tilde{\beta}_2)).$$

Then the components of $\tilde{M}_k$ fall under the same analysis as case b.

We have shown that if $\tilde{m}_{ij}$ is computed from the formulas given in part 2 a, b, c, d, and e using the computed quantities from part 1 and from the formation of $\tilde{D}$, then

$$fl(\tilde{m}_{ij}) = \tilde{m}_{ij} + \tilde{m}_{ij}\mu_{ij}(\varepsilon) + \nu_{ij}(\varepsilon) .$$

Define $\nu = \max_{i,k} |\nu_i^{(k)}|$. Then taking absolute value, using the triangle inequality, and recalling that $|\varepsilon_k| \le \varepsilon$ gives

$$|\mu_{ij}(\varepsilon)| \le \varepsilon(3+\varepsilon) ,$$

and

$$|\nu_{ij}(\varepsilon)| \le \begin{cases} \nu\varepsilon(1+\varepsilon) \text{ in case a,} \\ \nu(3\varepsilon(1 + \left|\frac{\delta_{21}}{\delta_{11}}\right|) + O(\varepsilon^2)) \text{ in case b,} \\ \nu\varepsilon(1+\varepsilon) \text{ in case c,} \\ \nu(\varepsilon(3 + 4\left|\frac{\delta_{21}}{\delta_{11}}\right| + 3\left|\frac{\delta_{31}}{\delta_{11}}\right|) + O(\varepsilon^2)) \text{ in case d,} \\ \nu(3\varepsilon(1 + \max(|\tilde{\delta}_{31}|, |\tilde{\delta}_{32}|) + O(\varepsilon^2)) \text{ in case e.} \end{cases}$$

By the properties of pivoting strategy S_α we have that

$$\left|\frac{\delta_{21}}{\delta_{11}}\right|, \left|\frac{\delta_{31}}{\delta_{11}}\right| < \frac{1}{\alpha}, \quad \text{and} \quad \max(|\tilde{\delta}_{31}|, |\tilde{\delta}_{32}|) \le \frac{1}{1-\alpha} .$$

Thus,

$$|\nu_{ij}(\varepsilon)| \leq \nu(\varepsilon(3 + \max[\frac{7}{\alpha}, \frac{1}{1-\alpha}]) + O(\varepsilon^2))$$

in all cases. □

We now return to equations (1.5). Regarding (1.5)(i) we have shown that $\Delta\tilde{M}_{ij} = \tilde{M}_{ij}\mu_{ij}(\varepsilon) + \nu_{ij}(\varepsilon)$, with bounds for $\mu_{ij}(\varepsilon)$ and $\nu_{ij}(\varepsilon)$ given in Lemma (3.4). We also have that $\Delta\tilde{D} = E$ in Lemma (3.3) is block diagonal with the same block structure as $\tilde{D}$. Using the analysis given in [3, p. 667] we see that in (1.5)(i) and (iv) that

$$|\delta_2\hat{M}_{ij}|, \ |\delta_1\hat{M}_{ij}| \leq \frac{3}{2}\varepsilon[1 + O(\varepsilon)](n-2+1-j)|\hat{M}_{ij}| \ ,$$

and in (1.5)(iii) we have

$$|\delta\hat{D}_{ii}| \leq |\hat{D}_{ii}|\varepsilon$$

if D_{ii} is a one-by-one block; otherwise, from [3]

$$\begin{pmatrix} |\delta\hat{D}_{ii}| & |(\delta\hat{D})_{i,i+1}| \\ |(\delta\hat{D})_{i,i+1}| & |(\delta\hat{D})_{i+1,i+1}| \end{pmatrix} \leq \varepsilon[1+O(\varepsilon)]|\hat{D}_{i+1,i}|\begin{pmatrix} \alpha & 1 \\ 1 & \alpha \end{pmatrix} .$$

Finally, we have that the error matrix S, defined in Lemma (3.2) and discussed in the remarks following it, is bounded by

$$\begin{aligned} \|S\|_\infty &\leq \{2\|T\|_\infty\|p\|_\infty\|w\|_\infty + \|T\|_\infty^2\|p\|_\infty^2\}|\sigma| \\ &\leq \{2n(n+1)\gamma \max|m_{ij}|\max|w_i^{(k)}| \ \|w\|_\infty\varepsilon \\ &\quad + \varepsilon^2[n(n+1)]^2\gamma^2\max|w_i^{(k)}|^2\}|\sigma| \ , \end{aligned}$$

where γ is the constant appearing in Lemma (3.2). The matrix $G(\varepsilon)$ in (1.7)(i) and (iii) is given by $G = (\nu_{ij}(\varepsilon))$. (Note that G is lower triangular.)

We have the bound for F given by

$$(3.12)\qquad \|F\|_\infty \le 2 \max_{i,k} |v_i^{(k)}| \varepsilon (3 + \max(\tfrac{7}{\alpha}, \tfrac{1}{1-\alpha})) \|\tilde{\tilde{M}}\tilde{\tilde{D}}\tilde{\tilde{M}}^t\|_\infty$$
$$+ 2 \max_{ijk} (|D_{ij}|, \alpha|\beta_k (v_1^{(k)})^2|, 2|\beta_k v_i^{(k)} \omega_j^{(k)}|,$$
$$|\sigma_k (\omega_i^{(k)})^2|) 49\varepsilon |\tilde{M}|\|_\infty \|\tilde{M}^t\|_\infty$$
$$+ \varepsilon[1 + O(\varepsilon)] \|MDM^t\|_\infty + \|S\|_\infty + O(\varepsilon^2) .$$

We have already mentioned that the $|v_i^{(k)}|$ and $|\omega_j^{(k)}|$, and $|S|_\infty$ may grow with n for ill-conditioned matrices M. However, the computational evidence indicates that the usual remarks concerning the solution of triangular systems apply: in practice large growth does not occur in these quantities.

In order to guarantee stability we must also show that the σ_k's and β_k's are bounded. We shall do this by showing that the growth of σ_k is bounded at each step of the algorithm. This is sufficient since it can be demonstrated that the growth of β_{k+1} is bounded as long as the growth of σ_{k+1} is bounded.

It will be necessary to impose an additional condition on the acceptance of a 1×1 pivot. The number δ in

$$\begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix}$$

will be accepted as a 1×1 pivot if $|\delta| \ge \alpha|\beta|$ as before, or if $|\sigma\delta| > \alpha\beta^2$. This does not affect any of the preceding analysis.

We shall begin by establishing several preliminary lemmas.

Lemma (3.5). Let A be symmetric and suppose that $A = MDM^t$. Let the eigenvalues of A be $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, and let the eigenvalues of D be $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$. Let k be the index such that $\lambda_j \leq 0$ for $1 \leq j \leq k$, and $\lambda_j > 0$ for $k < j < n$. Then $\lambda_j \leq c\mu_j$ for $1 \leq j \leq k$, and $\lambda_j \geq c\mu_j$ for $k < j \leq n$, where $\sqrt{c}$ is the smallest singular value of M.

Proof: By the mini-max theorem

$$\lambda_j = \min_{\dim(S)=j} \left[\max_{\substack{x \in S \\ x \neq 0}} \frac{x^t A x}{x^t x} \right]$$

$$= \min_{\dim(S)=j} \left[\max_{\substack{s \in S \\ s \neq 0}} \frac{s^t D s}{s^t M^{-1} M^{-t} s} \right]$$

$$= \min_{\dim(S)=j} \left[\max_{\substack{s \in S \\ s \neq 0}} \left(\frac{s^t D s}{s^t s} \right) \left(\frac{s^t s}{s^t M^{-1} M^{-t} s} \right) \right] .$$

Since $\frac{s^t s}{s^t M^{-1} M^{-t} s} \geq c > 0$, we have for $j \leq k$ that

$$\lambda_j \leq c \min_{\dim(S)=i} \left[\max_{\substack{s \in S \\ s \neq 0}} \frac{s^t D s}{s^t s} \right] = c\mu_j .$$

Similarly, for $j > k$ we have $\lambda_j \geq c\mu_j$. □

For the following discussion we shall also need to know the smallest singular value of certain lower triangular matrices in order to apply Lemma (3.5).

Lemma (3.6). If $M = \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix}$, then the smallest singular value of M is $\sqrt{c}$, where

$$c \geq \frac{1}{\gamma^2 + 2} .$$

If $M = \begin{pmatrix} 1 & 0 & 0 \\ \gamma_0 & 1 & 0 \\ \gamma_1 & 0 & 1 \end{pmatrix}$, then the smallest singular value of M is $\sqrt{c}$, where

$$c \geq \frac{1}{\gamma_0^2 + \gamma_1^2 + 2} .$$

Proof: The smallest singular value of $\begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix}$ is $\sqrt{c}$, where

$$c = (\gamma^2 + 2 - \sqrt{\gamma^4+4\gamma^2})/2 .$$

The smallest singular value of $\begin{pmatrix} 1 & 0 & 0 \\ \gamma_0 & 1 & 0 \\ \gamma_1 & 0 & 1 \end{pmatrix}$ is $\sqrt{c}$, where

$$c = \left(\gamma_0^2 + \gamma_1^2 + 2 - \sqrt{(\gamma_0^2+\gamma_1^2)^2 + 4(\gamma_0^2+\gamma_1^2)}\right)/2 .$$

If $c = (a + 2 - \sqrt{a^2 + 4a})/2$ with $a > 0$, then

$$c = \frac{(a+2)^2 - (a^2+4a)}{2(a + 2 + \sqrt{a^2+4a})}$$

$$= \frac{2}{a + 2 + \sqrt{a^2+4a}}$$

$$\geq \frac{2}{a + 2 + \sqrt{a^2+4a+4}}$$

$$= \frac{2}{2(a+2)} = \frac{1}{a+2} .$$

The lemma follows from this inequality. □

It will also be of interest in the following discussion to bound the ∞-norm of the inverse of a 2×2 block.

Lemma (3.7). Suppose that $D = \begin{pmatrix} \delta_{11} & \delta_{21} \\ \delta_{21} & \delta_{22} \end{pmatrix}$, with $\alpha|\delta_{21}| > \max(|\delta_{11}|, |\delta_{22}|)$. Then

$$\|D^{-1}\|_\infty \leq \frac{1}{(1-\alpha)|\delta_{21}|} .$$

Proof:

$$\|D^{-1}\|_\infty = \frac{1}{|\Delta|} \|D\|_\infty ,$$

where $|\Delta| = |\delta_{11}\delta_{22} - \delta_{21}^2| \geq \delta_{21}^2 - \alpha^2\delta_{21}^2 = \delta_{21}^2(1-\alpha^2)$.

$$\|D\|_\infty = |\delta_{21}| + \max(|\delta_{11}|, |\delta_{22}|)$$

$$\leq (1+\alpha)|\delta_{21}| .$$

Thus $\|D^{-1}\|_\infty \leq \frac{(1+\alpha)}{(1-\alpha^2)} \left(\frac{1}{|\delta_{21}|}\right) = \frac{1}{(1-\alpha)|\delta_{21}|}$. □

Two more lemmas are needed before we can establish the boundedness of the σ_k.

Lemma (3.8). Let $B \in R^{n\times n}$ be symmetric and nonsingular with eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, where $n \geq 2$. Let $z \in R^n$, and $\eta \in R$. Let

$$B' = B + \eta z z^t .$$

Then $n \max|B'_{ij}| \geq \lambda$, where $\lambda = \min_{1\leq j\leq n} |\lambda_j|$.

Proof: Let B' have eigenvalues $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$. From Lemma (2.2) of Chapter II,

$$\eta > 0 \Rightarrow \lambda_1 \leq \mu_1 \leq \lambda_n \leq \mu_n ,$$

while

$$\eta < 0 \Rightarrow \mu_1 \leq \lambda_1 \leq \mu_n \leq \lambda_n .$$

If $\lambda_1 > 0$, then $|\mu_n| \geq |\lambda_1|$.

If $\lambda_1 < 0$ and $\lambda_n > 0$, then $|\mu_n| \geq |\lambda_n|$ when $\eta > 0$ and $|\mu_1| \geq |\lambda_1|$ when $\eta < 0$.

If $\lambda_n < 0$, then $|\mu_1| \geq |\lambda_n|$.

We conclude that

$$\mu = \max_{1 \leq j \leq n} |\mu_j| \geq \lambda \ .$$

By the standard norm inequalities we obtain

$$n \max |B'_{ij}| \geq \mu$$

and the result is established. □

Lemma (3.9). Suppose that the k-th step of the updating algorithm has resulted in

$$C^{(k)} = \begin{pmatrix} 1 & 0 \\ v & w \end{pmatrix} \begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix} \begin{pmatrix} 1 & 0 \\ v & w \end{pmatrix}^t ,$$

with $\sigma \neq 0$, $|\delta| < \alpha|\beta|$ and $|\sigma\delta| \leq \alpha\beta^2$. Then

$$\beta^2 \geq \frac{|\sigma| \ |\det \hat{D}_1| \ |1 + \sigma_0 z^t A^{-1} z|}{(1+\alpha)|\det \tilde{D}^{(k)}| \left(|\sigma| |w^t \hat{M}_2^{-t} \hat{D}_2^{-1} \hat{M}_2^{-1} w| + 1 + \frac{1}{1-\alpha} \right)} ,$$

where $$M = (\hat{M}_1, \hat{M}_2), \quad D = \begin{pmatrix} \hat{D}_1 & 0 \\ 0 & \hat{D}_2 \end{pmatrix},$$

$$QAQ^t = (\hat{M}_1, \hat{M}_2) \begin{pmatrix} \hat{D}_1 & 0 \\ 0 & \hat{D}_2 \end{pmatrix} (\hat{M}_1, \hat{M}_2)^t ,$$

and $\tilde{D}^{(k)}$ is that portion of $\tilde{D}$ obtained up to step k. Here σ_0 is the starting "σ", and σ is the modified "σ" after k steps of the algorithm.

Proof: At the k-th step we have

$$\tilde{Q}^{(k)}\tilde{\tilde{A}}\tilde{\tilde{Q}}^{(k)t} = \tilde{M}^{(k)}\tilde{D}^{(k)}\tilde{M}^{(k)t} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v & w \end{pmatrix}\begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 1 & 0 \\ v & w \end{pmatrix}^t + \hat{M}_2\hat{D}_2\hat{M}_2^t$$

$$= \left(\begin{array}{c|c|c} \tilde{M}^{(k)} & \begin{matrix} 0 \\ 1 \\ v \end{matrix} & \hat{M}_2 \end{array}\right)\begin{pmatrix} \tilde{D}^{(k)} & 0_2 & 0 \\ 0 & (\delta-\beta^2/\sigma) & 0 \\ 0 & 0 & \hat{D}_2 \end{pmatrix}\left(\begin{array}{c|c|c} \tilde{M}^{(k)} & \begin{matrix} 0 \\ 1 \\ v \end{matrix} & \hat{M}_2 \end{array}\right)^t$$

$$+ \sigma\begin{pmatrix} 0 \\ \beta/\sigma \\ w+\beta/\sigma v \end{pmatrix}(0,\ \beta/\sigma,\ w^t + \beta/\sigma v^t) \ .$$

Now, $\det(A + \sigma_0 zz^t) = \det A(1 + \sigma_0 z^t A^{-1} z)$. Also, if $p = \begin{pmatrix} 0 \\ \beta/\sigma \\ w+\beta/\sigma v \end{pmatrix}$, and

$$B = \left(\begin{array}{c|c|c} M^{(k)} & \begin{matrix} 0 \\ 1 \\ v \end{matrix} & \hat{M}_2 \end{array}\right)\begin{pmatrix} \tilde{D}^{(k)} & 0_2 & 0 \\ 0 & (\delta-\beta^2/\sigma) & 0 \\ 0 & 0 & \hat{D}_2 \end{pmatrix}\left(\begin{array}{c|c|c} \tilde{M}^{(k)} & \begin{matrix} 0 \\ 1 \\ v \end{matrix} & \hat{M}_2 \end{array}\right) ,$$

then

$$\det A(1 + \sigma_0 z^t A^{-1} z) = \det(B + \sigma pp^t)$$

$$= \det B(1 + \sigma p^t B^{-1} p)$$

$$= \det \tilde{D}^{(k)}\left(\delta - \frac{\beta^2}{\sigma}\right)\det \hat{D}_2(1 + \sigma p^t B^{-1} p) \ .$$

Thus

$$(3.13) \quad |\det A||1 + \sigma_0 z^t A^{-1} z| = |\det \tilde{D}^{(k)}||\det \hat{D}_2||\delta - \frac{\beta^2}{\sigma}||1 + \sigma p^t B^{-1} p| .$$

Since $|\delta\sigma| \leq \alpha\beta^2$, we have

$$(3.14) \qquad |\delta\sigma-\beta^2| \geq \beta^2 - |\delta\sigma| \geq (1-\alpha)\beta^2 \ ,$$

and

$$(3.15) \qquad |\delta\sigma-\beta^2| \leq |\delta\sigma| + \beta^2 \leq (\alpha+1)\beta^2 \ .$$

Also, one can show that

$$\begin{pmatrix} M^{(k)} & \begin{matrix} 0 \\ 1 \\ v \end{matrix} & \\ & & \hat{M}_2 \end{pmatrix}^{-1} p = \begin{pmatrix} 0 \\ \beta/\sigma \\ -\beta/\sigma \hat{M}_2^{-1} v + \hat{M}_2^{-1}(w+\beta/\sigma v) \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ \beta/\sigma \\ \hat{M}_2^{-1} w \end{pmatrix},$$

where we use $\hat{M}_2^{-1}$ to mean the inverse of the unit lower triangular matrix that occupies the lower triangle of $\hat{M}_2$. Thus if $\phi = w^t \hat{M}_2^{-t} \hat{D}_2^{-1} \hat{M}_2^{-1} w$, we have

$$p^t B^{-1} p = \frac{\beta^2}{\sigma^2} \frac{1}{\left(\delta - \frac{\beta^2}{\sigma}\right)} + \phi \ .$$

Hence

$$(3.16) \qquad |p^t B^{-1} p| \leq \frac{\beta^2}{|\sigma||\sigma\delta - \beta^2|} + |\phi|$$

$$\leq \frac{1}{|\sigma|(1-\alpha)} + |\phi|$$

by (3.14). Using (3.15) in (3.13) gives

$$(3.17) \qquad \frac{1}{|\sigma|}(1+\alpha)\beta^2 \geq |\delta - \beta^2/\sigma| = \frac{|\det A||1 + \sigma_0 z^t A^{-1} z|}{|\det D^{(k)}||\det \hat{D}_2||1 + \sigma p^t B^{-1} p|} \ .$$

Therefore, using (3.16) in (3.17) gives

$$(3.18) \qquad \beta^2 \geq \frac{|\sigma|}{(1+\alpha)} \frac{|\det \hat{D}_1||1 + \sigma_0 z^t A^{-1} z|}{|\det \tilde{D}^{(k)}|(|\sigma||\phi| + 1 + \frac{1}{1-\alpha})} \ .$$

This is the desired result. □

Observe that the quantities in (3.18) are independent of the updating process except for $|\sigma|$, and $|\det \tilde{D}^{(k)}|$. Now we are ready to prove

Theorem (3.1). Suppose that the k-th step of the updating algorithm has resulted in

$$C^{(k)} = \begin{pmatrix} 1 & 0 \\ v & w \end{pmatrix} \begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix} \begin{pmatrix} 1 & 0 \\ v & w \end{pmatrix}^t,$$

with $|\delta| < \alpha|\beta|$, and $|\sigma\delta| \leq \alpha\beta^2$. Then the next step of the algorithm will produce a σ' of the form

$$\sigma' = \sigma - b_k^t \tilde{D}_k^{-1} b_k$$

with $|\sigma'|$ bounded.

Proof: Let $\begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix} \begin{pmatrix} 1 \\ \gamma \end{pmatrix} = \mu_1 \begin{pmatrix} 1 \\ \gamma \end{pmatrix}$, and $\begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix} \begin{pmatrix} -\gamma \\ 1 \end{pmatrix} = \mu_2 \begin{pmatrix} -\gamma \\ 1 \end{pmatrix}$ with $|\mu_1| \geq |\mu_2|$ as in Lemma (2.4) of Chapter II. Let $\eta_j = \mu_j/(1+\gamma^2)$, j=1,2.

As we have already seen in Chapter II the updating process is equivalent to forming

$$C^{(k+1)} = \begin{pmatrix} 1 & 0 & w_0 \\ v_1 & I & w_1 \\ v_2 & V & w_2 \end{pmatrix} \begin{pmatrix} \eta_1 & 0 & 0 \\ 0 & D' & 0 \\ 0 & 0 & \eta_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & w_0 \\ v_1 & I & w_1 \\ v_2 & V & w_2 \end{pmatrix}^t, \tag{3.19}$$

where D' and $\begin{pmatrix} 0 \\ I \\ V \end{pmatrix}$ are, respectively, the next diagonal block of D and the corresponding column of M. In (3.19) $w_0 = -\gamma$, $\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = v+\gamma w$, and $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = w-\gamma v$.

The next step is to form

$$C^{(k+1)} = \left(\begin{array}{cc|c} 1 & 0 & w_0 \\ 0 & I & w_1 \\ \hat{v}_2 & V & w_2 \end{array}\right) \left(\begin{array}{cc|c} \hat{D} & & 0 \\ & & \\ 0 & & \eta_2 \end{array}\right) \left(\begin{array}{cc|c} 1 & 0 & w_0 \\ 0 & I & w_1 \\ \hat{v}_2 & V & w_2 \end{array}\right)^t,$$

where $\hat{D} = \begin{pmatrix} 1 & 0 \\ v_1 & I \end{pmatrix} \begin{pmatrix} \eta_2 & 0 \\ 0 & D' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ v_1 & I \end{pmatrix}^t$, and $\hat{v}_2 = v_2 - V v_1$.

Then we form

$$C^{(k+1)} = \left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & I & 0 \\ \hat{v}_2 & V & \hat{w}_2 \end{array}\right)\left(\begin{array}{c|c} \hat{D} + \eta_2 bb^t & \eta_2 b \\ \hline \eta_2 b^t & \eta_2 \end{array}\right)\left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & I & 0 \\ \hat{v}_2 & V & \hat{w}_2 \end{array}\right)^t ,$$

where $b = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$, and $\hat{w}_2 = w_2 - \hat{v}_2 w_0 - V w_1$.

Finally, we pivot and obtain the updated diagonal block $\tilde{D}_{k+1}$. Let $\bar{D} = \hat{D} + \eta_2 bb^t$, and let $\xi = \max|\bar{D}_{ij}|$. Then $|\tilde{D}_k| \geq \alpha\xi$ when $\tilde{D}_{k+1}$ is 1×1. If it is 2×2, $\|\tilde{D}_{k+1}^{-1}\|_\infty \leq \frac{1}{\xi(1-\alpha)}$. To avoid cumbersome notation we shall let $B = \tilde{D}_{k+1}$.

The factoring of $\begin{pmatrix} \bar{D} & \eta_2 b \\ \eta_2 b^t & \eta_2 \end{pmatrix}$ gives an updated $\eta = \eta_2 - \eta_2^2 b_1^t B^{-1} b_1$, where b_1 consists of components of b. It can be shown that $\hat{w}_2 = (1+\gamma^2)(w_2' - V w_1')$, where $w = \begin{pmatrix} w_1' \\ w_2' \end{pmatrix}$. Comparing this with the updating algorithm described in Chapter II will show that

$$\sigma' = (1+\gamma^2)^2 \eta \ . \tag{3.20}$$

Now,

$$\begin{aligned} |\eta| &\leq |\eta_2| + \eta_2^2 |b_1^t B^{-1} b_1| \\ &\leq |\eta_2| + 3\eta_2^2 \|b_1\|_\infty^2 \|B^{-1}\|_\infty \ . \end{aligned}$$

Hence, if $\alpha \geq 1/2$ then

$$|\eta| \leq |\eta_2| + \frac{3\eta_2^2 \|b_1\|_\infty^2}{\xi(1-\alpha)} \ . \tag{3.21}$$

Let $\sqrt{\theta}$ be the smallest singular value of $\begin{pmatrix} 1 & 0 \\ v_1 & I \end{pmatrix}$. From Lemma (3.6),

$$\theta \geq 1/(\|v_1\|^2 + 2) \ .$$

By Lemma (3.8) we have that $3\xi \geq |\lambda|$ where λ is the smallest eigenvalue of $\hat{D}$ (in absolute value). However, Lemma (3.5) implies $|\lambda| \geq \theta \min(|\eta_1|, |\mu|)$,

and thus

$$\xi \geq \frac{1}{3}\,\theta\,\min(|\eta_1|,|\mu|) \tag{3.23}$$

where μ is the smallest eigenvalue of D' (in absolute value). Finally we have that

$$\nu \equiv \|b_1\|_\infty \leq \|b\|_\infty \leq \max(|\gamma|,\ \|w\|_\infty + |\gamma|\ \|v\|_\infty)\ . \tag{3.24}$$

Combining inequalities (3.21)-(3.24) gives

$$|\eta| \leq |\eta_2| + \frac{\nu^2 9\eta_2^2(\|v_1\|^2+2)}{\min(|\eta_1|,|\mu|)(1-\alpha)}\ . \tag{3.25}$$

Since $\|w\|_\infty$ and $\|v\|_\infty$ are bounded we have that $\nu^2 = O(\gamma^2)$ and $\|v_1\|^2 = O(\gamma^2)$. Thus it is sufficient to bound the quantity

$$\left| \frac{\eta_2\gamma^4}{\min(|\eta_1|,|\mu|)} \right|\ . \tag{3.26}$$

Suppose that $|\beta| \geq |\sigma|$. Since $\gamma = (\sigma-\mu_2)/\beta$ we have

$$|\gamma| \leq \left| \frac{(\sigma+\delta) - \operatorname{sgn}(\sigma+\delta)\sqrt{(\sigma-\delta)^2 + 4\beta^2}}{2\beta} \right| + 1$$

$$\leq \frac{1}{2}\left(1 + \alpha + \sqrt{(1+\alpha)^2 + 4}\right) + 1\ .$$

Thus (3.26) is bounded since $\left|\frac{\eta_2}{\eta_1}\right| \leq 1$ and $|\mu|$ is fixed. Therefore, we shall assume that $|\beta| < |\sigma|$. Now,

$$|\eta_2| = \left|\frac{\delta\sigma - \beta^2}{\eta_1}\right| \left(\frac{1}{1+\gamma^2}\right)^2\ .$$

However,

$$\left|\frac{\delta\sigma - \beta^2}{\eta_1}\right| = \mu_2(1+\gamma^2) = \mu_2 + \mu_2\gamma^2\ .$$

Let $\psi = \operatorname{sgn}(\sigma+\delta)\sqrt{(\sigma-\delta)^2+4\beta^2}$. Then

$$\frac{\gamma^2\mu_2}{\sigma} = \frac{[(\sigma-\delta)+\psi]^2[(\sigma-\delta)-\psi+2\delta]}{8\beta^2\sigma}$$

$$= \left(\frac{(\sigma-\delta)+\psi}{2\sigma}\right)\left(\frac{(\sigma-\delta)^2-\psi^2}{4\beta^2}\right) + \frac{2\delta[(\sigma-\delta)+\psi]^2}{8\beta^2\sigma} .$$

Thus

$$\left|\frac{\gamma^2\mu_2}{\sigma}\right| \leq \tau + \left|\frac{\delta\sigma}{\beta^2}\right|\tau^2 ,$$

where $\tau = \left|\frac{(\sigma-\delta)+\psi}{2\sigma}\right|$. Observe that

$$\tau \leq \frac{1}{2}\left[1 + \left|\frac{\delta}{\sigma}\right| + \sqrt{(1+\left|\frac{\delta}{\sigma}\right|)^2 + 4\left|\frac{\beta}{\sigma}\right|^2}\right]$$

$$\leq \frac{1}{2}\left[1 + \alpha + \sqrt{(1+\alpha)^2+4}\right] .$$

By assumption $\left|\frac{\delta\sigma}{\beta^2}\right| \leq \alpha < 1$ so that

$$\left|\frac{\gamma^2\mu_2}{\sigma}\right| \leq \tau + \tau^2 .$$

We also have that

$$|\mu_2| = \frac{1}{2}\left|[(\sigma+\delta) - \psi]\right| = \frac{1}{2}\left|[(\sigma-\delta) - \psi + 2\delta]\right|$$

$$\leq \frac{1}{2}\frac{4\beta^2}{|(\sigma-\delta)+\psi|} + |\delta| .$$

Now,

$$|(\sigma-\delta)+\psi| = |\sigma-\delta| + \sqrt{|\sigma-\delta|^2 + 4\beta^2} \geq 2|\sigma-\delta| .$$

Thus,

$$|\mu_2| \leq \frac{\beta^2}{|\sigma-\delta|} + |\delta| \leq \frac{|\beta|}{(1-\alpha)} + |\delta| .$$

Therefore,

$$|\eta_2| \leq (1+\gamma^2)^{-2}\left(\frac{|\beta|}{1-\alpha} + |\delta| + |\sigma|(\tau+\tau^2)\right) .$$

It follows that

$$\left|\frac{\gamma^4\eta_2}{\mu}\right| \leq \frac{1}{\mu}\left(\frac{|\beta|}{1-\alpha} + |\delta| + |\sigma|(\tau+\tau^2)\right)$$

$$\leq \frac{|\sigma|}{\mu}\left(\frac{1}{1-\alpha} + \alpha + \tau + \tau^2\right) .$$

To bound $\left|\frac{\eta_2\gamma^4}{\eta_1}\right|$ we consider

$$\left|\frac{\eta_2}{\eta_1}\right| = \left|\frac{\delta\sigma - \beta^2}{\mu_1^2}\right| \leq \frac{(1+\alpha)\beta^2}{\mu_1^2} .$$

But

$$\mu_1^2 = (\sigma+\delta)^2 + 2|\sigma+\delta|\sqrt{(\sigma-\delta)^2 + 4\beta^2} + (\sigma-\delta)^2 + 4\beta^2$$

$$\geq 2\sigma^2 + 2\delta^2 + 2(\sigma^2-\delta^2) + (\sigma-\delta)^2 + 4\beta^2$$

$$= 5\sigma^2 - 2\sigma\delta + \delta^2 + 4\beta^2$$

$$\geq 5\sigma^2 + (4-2\alpha)\beta^2 \geq 5\sigma^2 .$$

Thus

$$\left|\frac{\eta_2}{\eta_1}\right| \leq \frac{(1+\alpha)}{5}\frac{\beta^2}{\sigma^2} .$$

But

$$\gamma^2 = \left(\frac{\mu_1-\delta}{\beta}\right)^2 = \frac{\sigma^2}{\beta^2}\left[\frac{(\sigma-\delta) + \psi}{2\sigma}\right]^2$$

$$\leq \frac{\sigma^2}{4\beta^2}\left[1 + \alpha + \sqrt{(1+\alpha)^2 + 4}\right]^2 .$$

It follows that

$$\left|\frac{\gamma^4\eta_2}{\eta_1}\right| \le \frac{\sigma^4}{16\beta^4}\left|\frac{\eta_2}{\eta_1}\right|\left[1+\alpha+\sqrt{(1+\alpha)^2+4}\right]^4$$

$$\le \frac{\sigma^2}{80\beta^2}(1+\alpha)\left[1+\alpha+\sqrt{(1+\alpha)^2+4}\right]^4$$

Using the same notation as in Lemma (3.9) and applying that result gives

$$\frac{\sigma^2}{\beta^2} \le \frac{(1+\alpha)|\sigma||\det \tilde{D}^{(k)}|\left(|\sigma||\phi| + 1 + \frac{1}{1-\alpha}\right)}{|\det \hat{D}_1|\ |1+\sigma_0 z^t A^{-1} z|} .$$

Observe that if the previous σ's and β's are bounded then $|\det \tilde{D}^{(k)}|$ is bounded. Thus we have bounds

$$\left|\frac{\gamma^4\eta_2}{\mu}\right| \le K_1 \text{ and } \left|\frac{\gamma^4\eta_2}{\eta_1}\right| \le K_2 .$$

This gives

$$|\eta| \le |\eta_2|(1+\max(K_1,K_2)) .$$

But

$$|\eta_2| \le |\sigma|(1+\gamma^2)^{-2}\left(\frac{1}{1-\alpha}+\alpha+\tau+\tau^2\right) .$$

Since $\sigma' = (1+\gamma^2)^2\eta_2$ we have

$$|\sigma'| \le |\sigma|\left(\frac{1}{1-\alpha}+\alpha+\tau+\tau^2\right)(1+\max(K_1,K_2)) .$$

This concludes the proof. □

Theorem (3.1) provides a bound on the growth of σ when pivoting is done. The following theorem provides a bound in the remaining cases.

Theorem (3.2). If the algorithm updates a 2×2 block and accepts the updated block as a 2×2 pivot then

$$|\sigma'| \leq |\sigma|\left(1 + \frac{2}{1-\alpha}\right)$$

or if a 1x1 pivot is accepted from the updated block then

$$|\sigma'| \leq |\sigma|\left(1 + \frac{4\alpha}{1-\alpha}\right) .$$

If δ is accepted from $\begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix}$ as a 1x1 block then

$$|\sigma'| \leq |\sigma| + \max(|\sigma|,|\beta|)/\alpha .$$

Proof:

Case 1. If δ is accepted as a 1x1 block as a result of

$$\begin{pmatrix} \delta & \beta \\ \beta & \sigma \end{pmatrix}$$

satisfying $|\delta| \geq \alpha|\beta|$, or $|\delta\sigma| > \alpha\beta^2$ then

$$\sigma' = \sigma - \beta^2/\delta .$$

Hence
$$|\sigma'| \leq |\sigma| + \left|\frac{\beta^2}{\delta}\right|$$
$$\leq |\sigma| + |\sigma|/\alpha = |\sigma|(1 + 1/\alpha) ,$$

or
$$|\sigma'| \leq |\sigma| + |\beta|/\alpha .$$

Case 2. Suppose that a 2x2 block is updated and accepted as a 2x2 pivot. If the old block has elements δ_{ij} then the updated block has elements of the form $\tilde{\delta}_{ij} = \delta_{ij} + \sigma w_i w_j$. The conditions that must be satisfied are $\alpha|\delta_{21}| > \max(|\delta_{11}|,|\delta_{22}|)$, and $\alpha|\tilde{\delta}_{21}| > \max(|\tilde{\delta}_{11}|,|\tilde{\delta}_{22}|)$. Let $\tilde{D}$ represent this 2x2 updated block, and let $w^t = (w_1,w_2)$. Without loss of generality we assume $\|w\|_\infty = |w_1|$. Then $\|w^t\|_\infty \leq 2|w_1|$. In this case we have

$$\sigma' = \sigma - \sigma^2 w^t \tilde{D}^{-1} w$$

so that

$$|\sigma'| \leq |\sigma|(1 + 2|\sigma||w_1|^2\|\tilde{D}^{-1}\|_\infty) .$$

From Lemma (3.7), $\|\tilde{D}^{-1}\|_\infty \leq \frac{1}{(1-\alpha)|\tilde{\delta}_{21}|}$.

Now,

$$|\sigma w_1^2| - |\delta_{11}| \leq |\delta_{11}+\sigma w_1^2| < \alpha|\delta_{21}+\sigma w_2 w_1| .$$

Thus

$$\begin{aligned} |\sigma w_1^2| &\leq \alpha|\tilde{\delta}_{21}| + |\delta_{11}| \\ &\leq \alpha(|\tilde{\delta}_{21}| + |\delta_{21}|) \\ &\leq \alpha(|\tilde{\delta}_{21}| + |\delta_{21} + \sigma w_2 w_1| + |\sigma w_2 w_1|) \\ &\leq 2\alpha|\tilde{\delta}_{21}| + \alpha|\sigma w_1^2| . \end{aligned}$$

Hence $(1-\alpha)|\sigma w_1^2| \leq 2\alpha|\delta_{21}|$, and we have

$$\begin{aligned} |\sigma'| &\leq |\sigma|\left(1 + 2\left(\frac{2\alpha}{1-\alpha}\right)\frac{|\sigma w_1^2|}{|\sigma w_1^2|}\right) \\ &= |\sigma|\left(1 + \frac{4\alpha}{1-\alpha}\right) \end{aligned}$$

Case 3. A 2×2 block is updated and it is found that $\alpha|\tilde{\delta}_{21}| \leq \max(|\tilde{\delta}_{11}|, |\tilde{\delta}_{22}|)$. We use the same notation as for Case 2. Without loss of generality we assume that $|\tilde{\delta}_{11}| \geq |\tilde{\delta}_{22}|$ (otherwise $\tilde{\delta}_{22}$ is brought to the $\tilde{\delta}_{11}$ position). Then

$$\sigma' = \sigma - \frac{\sigma^2 w_1^2}{\tilde{\delta}_{11}} .$$

Now

$$|\delta_{11}+\sigma w_1^2| \geq |\sigma w_1^2| - |\delta_{11}|$$

so that

$$|\sigma w_1^2| \leq |\tilde{\delta}_{11}| + |\delta_{11}| \ .$$

Thus

$$\frac{|\sigma w_1^2|}{|\tilde{\delta}_{11}|} \leq 1 + \frac{|\delta_{11}|}{|\tilde{\delta}_{11}|} \ .$$

But

$$\begin{aligned} |\delta_{11}| &\leq \alpha|\delta_{21}| \\ &\leq \alpha(|\delta_{21}+\sigma w_2 w_1| + |\sigma w_2 w_1|) \\ &= \alpha(|\tilde{\delta}_{21}| + |\sigma w_2 w_1|) \ . \end{aligned}$$

Thus

$$\frac{|\sigma w_1^2| - \alpha|\sigma w_2 w_1|}{|\tilde{\delta}_{11}|} \leq 1 + \alpha \frac{|\tilde{\delta}_{21}|}{|\tilde{\delta}_{11}|} \leq 2 \ . \tag{3.27}$$

A similar argument shows that

$$\frac{|\sigma w_2^2|}{|\tilde{\delta}_{11}|} \leq \frac{|\tilde{\delta}_{22}| + |\delta_{22}|}{|\tilde{\delta}_{11}|} \leq \left[1 + \alpha\left(\frac{|\tilde{\delta}_{21}| + |\sigma w_2 w_1|}{|\tilde{\delta}_{11}|}\right)\right].$$

Thus

$$\frac{|\sigma w_2^2| - \alpha|\sigma w_2 w_1|}{|\tilde{\delta}_{11}|} \leq 2 \ . \tag{3.28}$$

If $|w_1| \geq |w_2|$ then $(1-\alpha)|\sigma w_1^2| \leq |\sigma w_1^2| - \alpha|\sigma w_1 w_2|$, and inequality (3.27) shows that

$$\frac{|\sigma w_1^2|}{|\tilde{\delta}_{11}|} \leq \frac{2}{1-\alpha} \ . \tag{3.29}$$

However, if $|w_2| > |w_1|$ then $(1-\alpha)|\sigma w_1^2| \leq (1-\alpha)|\sigma w_2^2| \leq |\sigma w_2^2| - \alpha|\sigma w_1 w_2|$,

and inequality (3.28) gives inequality (3.29). □

Theorem (3.1) shows that the growth of σ can be sensitive to near singularity in $\tilde{A}$. This can result in two ways. If σ is much larger than the eigenvalues of $\tilde{A}$ then numerically $\tilde{A}$ appears to be a rank one matrix. Also, one of the updated eigenvalues can be shifted to zero. This is reflected in the bounds obtained in Theorem (3.1) since one of the bounds depends on $\frac{\sigma}{\mu}$ where μ is an eigenvalue of D, and the other bound depends upon $1/(1+\sigma z^t A^{-1} z)$. The quantity $1+\sigma z^t A^{-1} z = 0$ if and only if $\tilde{A}$ has a zero eigenvalue. We conclude that the use of the algorithm should be restricted to cases where the matrices involved are well conditioned. Finally, we do not expect this technique to generalize to the LU decomposition of non-symmetric matrices since our results are heavily dependent upon properties of symmetric matrices.

Chapter IV

The Use of Directions of Negative Curvature in a Modified Newton Iteration

1. Introduction

In this chapter we present an algorithm for obtaining a numerical approximation to the solution of the following problem:

(1.1) Let $f: \mathcal{D} \subset R^n \to R$;

find $x^* \in \mathcal{D}$ such that

$$f(x^*) \leq f(x)$$

for all x in some neighborhood of x^*.

For theoretical reasons we shall assume once and for all that f has two continuous derivatives on $\mathcal{D}$ and that for any $x_0 \in \mathcal{D}$, the level set $L(x_0) = \{x: f(x) \leq f(x_0)\}$ is a compact subset of $\mathcal{D}$. Additional assumptions will be introduced as they are needed. The assumptions just stated shall be referred to as assumptions (1.2).

Recall from Chapter I that we denote the gradient of $f(x)$ by $g(x)$, and the Hessian by $G(x)$. Given a sequence of vectors $\{x_k\} \subset \mathcal{D}$ we shall use the notation $f_k = f(x_k)$, $g_k = g(x_k)$, and $G_k = G(x_k)$. We shall sometimes omit the argument x and write f for $f(x)$, and g for $g(x)$, etc., when there is no danger of confusion. Throughout this chapter we use $\|\cdot\|$ to denote the Euclidian norm, and $x^t y$ to denote inner products.

The algorithm we shall present may be classified as a descent method. Usually a descent method determines a descent direction s_k at the iterate x_k (i.e. $g_k^t s_k < 0$). Then a linear search is performed to obtain $\alpha_k > 0$ such that $f(x_k+\alpha_k s_k) \leq f_k$ and we take $x_{k+1} = x_k+\alpha_k s_k$.

Under some additional restrictions on the choice of α_k one can show that $\lim_{k\to\infty} g_k^t s_k / \|s_k\| = 0$. The vector s_k is usually related to g_k in such a way that this limit equalling zero implies that the iterates converge to a point x^* where $g(x^*) = 0$.

In addition to obtaining such a point x^* we would like to be able to assert that $G(x^*)$ is positive definite for this would imply that $f(x^*) < f(x)$ for all x in some neighborhood of x^*. Of course, we shall not be able to accomplish this goal, but through the use of directions of negative curvature we shall be able to guarantee that $G(x^*)$ is positive semidefinite. For practical purposes this is very strong assertion. For instance, if the Hessian were known to be nonsingular at all critical points then the point x^* would have to be a local minimum.

Recently the idea of using directions of negative curvature has appeared in modified Newton algorithms [8,11,16]. In particular we are indebted to the paper of McCormick [16]. Indeed, Theorem (3.1) is only a slight modification of McCormick's result. However, this result led us to consider a new line search strategy. The implementation of this strategy which we present here is based in a fundamental way on the factorization of symmetric matrices using the algorithm of Bunch and Parlett [5] and this is discussed in section 4. In section 5 we give termination criteria for the new univariate search strategy, and show how this relates to previous strategies. Finally, in section 6 we give a convergence result that includes various choices of descent directions and we suggest a particular way to define a modified Newton iteration.

Since the algorithm is a descent method we shall begin with a discussion of descent directions.

2. Descent Directions

The following definitions will be useful throughout this chapter.

Definition (2.1). Let $f: R^n \to R$ be twice differentiable in the open set $\mathcal{D}$.

(a) A point x in $\mathcal{D}$ is an indefinite point if G(x) has at least one negative eigenvalue.

(b) If x is an indefinite point then d is a direction of negative curvature if $d^tG(x)d < 0$.

(c) A pair of vectors (s,d) is a descent pair at the point x if when x is not an indefinite point then $g^ts < 0$, $g^td \leq 0$, and $d^tGd = 0$, while if x is an indefinite point then $g^ts \leq 0$, $g^td \leq 0$, and $d^tGd < 0$.

An example of a descent pair would be to take $s = -g(x)$. Then if G(x) is positive semidefinite take $d = 0$, and otherwise take $d = -\text{sgn}(g^te)e$ where e is the unit eigenvector corresponding to the most negative eigenvalue of G(x). We shall see that there are more attractive choices than this. However, regardless of the specific choice, a descent pair fails to exist at x only if $g(x) = 0$, and G(x) is positive semidefinite.

The search strategy we shall present departs from the usual strategy discussed in the introduction. Instead of using only one descent direction and searching in a line determined by that direction, we shall consider searching along a curve of the form

$$C: \{x_\alpha = x+\phi_1(\alpha)s+\phi_2(\alpha)d: \alpha \geq 0\} \tag{2.1}$$

with (s,d) a descent pair at x, and with $\phi_1(0) = \phi_2(0) = 0$. We hope to produce an $\bar{\alpha} > 0$ such that

$$f(x_\alpha) \leq f(x) \quad . \tag{2.2}$$

If we let $\Phi(\alpha) = f(x_\alpha)$ we encounter a univariate minimization problem where Φ'' is continuous as long as ϕ_1'', ϕ_2'' are continuous. The following lemma gives sufficient conditions under which (2.2) can be satisfied.

Lemma (2.1). Let $\Phi: R \to R$ be twice continuously differentiable on the open interval I which contains the origin, and assume that $\mu \in [0,1)$. Then there is an $\bar{\alpha} > 0$ such that

$$\Phi(\alpha) \leq \Phi(0) + \mu\left[\Phi'(0)\alpha + \Phi''(0)\frac{\alpha^2}{2}\right]$$

for all $\alpha \in [0,\bar{\alpha}]$ provided that either $\Phi'(0) < 0$, or $\Phi'(0) = 0$ and $\Phi''(0) < 0$.

Proof: The mean value theorem implies that for every $\upsilon > 0$ there exists $\theta \in (0,\alpha)$ such that

$$\Phi(\alpha) = \Phi(0) + \Phi'(0)\alpha + \Phi''(0)\frac{\alpha^2}{2} + \frac{1}{2}[\Phi''(\theta) - \Phi''(0)]\alpha^2 \quad .$$

Hence,

$$\Phi(\alpha) = \Phi(0) + \mu\left[\Phi'(0)\alpha + \Phi''(0)\frac{\alpha^2}{2}\right] + r(\alpha) \quad ,$$

where

$$r(\alpha) = (1-\mu)\left[\Phi'(0)\alpha + \Phi''(0)\frac{\alpha^2}{2}\right] + \frac{1}{2}[\Phi''(\theta) - \Phi''(0)]\alpha^2 \quad .$$

Since

$$\overline{\lim_{\alpha \to 0^+}} \frac{r(\alpha)}{\alpha^2} < 0 \quad ,$$

there exists an $\bar{\alpha} > 0$ such that $r(\alpha) < 0$ for all $\alpha \in [0,\bar{\alpha}]$. □

This lemma not only tells us when (2.2) can be satisfied, but also that the function f must decrease by a significant amount along the curve x_α. It also indicates that a larger decrease is likely when $\Phi''(0) < 0$. We, of course, want to use the simplest functions ϕ_1 and ϕ_2 which will guarantee that the hypothesis of Lemma (2.1) is satisfied. Observe that if $\Phi(\alpha) = f(x_\alpha)$ with x_α as in (2.1) then

$$\Phi'(0) = g(x)^t(\phi_1'(0)s + \phi_2'(0)d) , \tag{2.3}$$

$$\Phi''(0) = g(x)^t(\phi_1''(0)s + \phi_2''(0)d) + (\phi_1'(0)s + \phi_2'(0)d)^t G(x)(\phi_1'(0)s + \phi_2'(0)d) . \tag{2.4}$$

Suppose that $g^ts = g^td = 0$ at an indefinite point (this occurs for instance at a saddle point). Then in order to insure $\Phi''(0) < 0$ without imposing further conditions on s we must require $\phi_1'(0) = 0$, and $\phi_2'(0) > 0$. Then (2.3) and (2.4) simplify to

$$\Phi'(0) = g(x)^t(\phi_2'(0)d) , \tag{2.5}$$

$$\Phi''(0) = g(x)^t(\phi_1''(0)s + \phi_2''(0)d) + (\phi_2'(0)d)^t G(x)(\phi_2'(0)d) . \tag{2.6}$$

When G(x) is positive definite then d = 0 must be satisfied in order for (s,d) to be a descent pair. Thus $\Phi'(0) = 0$ and we must have $\phi_1''(0) > 0$ in order to insure $\Phi''(0) < 0$. Therefore, if $\phi_1(\alpha) = \sum_{j=0}^{\infty} \beta_j \alpha^j$ and $\phi_2(\alpha) = \sum_{j=0}^{\infty} \gamma_j \alpha^j$ then we must have $\beta_0 = \beta_1 = 0$ with $\beta_2 > 0$ and $\gamma_0 = 0$ with $\gamma_1 > 0$. The simplest functions of this type are, of course,

$$\phi_1(\alpha) = \alpha^2, \qquad \phi_2(\alpha) = \alpha .$$

In this case

$$\Phi'(0) = g(x)^t d \ , \tag{2.7}$$

$$\Phi''(0) = 2g(x)^t s + d^t G(x) d \ . \tag{2.8}$$

3. A Modification of the Armijo Steplength Procedure

In Section 2 we introduced the notion of a descent pair. The motivation for considering the use of a pair of vectors rather than the simpler strategy of determining a single direction of descent will be discussed now. We shall present here a modification of a theorem of McCormick. In [16] McCormick gives a modification of the Armijo steplength algorithm [2] which includes second derivative information in the form of directions of negative curvature.

The steplength algorithm will be described now. Given $\gamma, \mu \in (0,1)$, let $\{x_k: k=0,1,2,\ldots\}$ be a sequence of points derived from the given point x_0 as follows:

Determine a descent pair (s_k, d_k) at x_k and let i_k be the smallest non-negative integer i such that

$$y_{k,i} \equiv x_k + \gamma^{2i} s_k + \gamma^i d_k \in \mathcal{D} \tag{3.1}$$

and

$$f(y_{k,i}) \leq f_k + \mu\gamma^{2i}[g_k^t s_k + \tfrac{1}{2} d_k^t G_k d_k] \ . \tag{3.2}$$

Take $x_{k+1} = y_{k,i}$. Lemma (2.1) shows that the iterates are well defined, and if a descent pair does not exist at x_k then we accept x_k as a solution to problem (1.1).

Theorem (3.1). Let f satisfy assumptions (1.2) and suppose that $\|s_k\|, \|d_k\|$ are bounded independent of k. Then

(3.3) $$\lim_{k\to\infty} (-g_k^t s_k) = 0$$

and

(3.4) $$\lim_{k\to\infty} (-d_k^t G_k d_k) = 0 \ .$$

Proof: The sequence $\{f_k\}$ is a decreasing sequence which is bounded below due to the continuity of f and the compactness of $L(x_0)$. Thus $\lim_{k\to\infty} (f_k - f_{k+1}) = 0$. There are two cases to consider.

Case 1. Suppose the integers $\{i_k\}$ are bounded above by some $m \geq 0$. Then

$$f_k - f_{k-1} \leq -\mu\gamma^{2m}\left[g_k^t s_k + \frac{1}{2} d_k^t G_k d_k\right] \ .$$

Since

$$-g_k^t s_k \geq 0 \quad \text{and} \quad -d_k^t G_k d_k \geq 0$$

the conclusion follows.

Case 2. The integers $\{i_k\}$ are not bounded above. Without loss of generality we assume that $\lim_{k\to\infty} i_k = +\infty$. By the definition of i_k, if $\sigma_k = \gamma^{(i_k - 1)}$, then

(3.5) $$f_{k+1} \leq f_k + \mu\sigma_k^2\left[g_k^t s_k + \frac{1}{2} d_k^t G_k d_k\right] \ .$$

However, due to our assumptions on f and $L(x_0)$, a Taylor series argument and the fact that $g_k^t d_k \leq 0$ may be used to show that

(3.6) $$f_{k+1} \leq f_k + \sigma_k^2\left[g_k^t s_k + \frac{1}{2} d_k^t G_k d_k\right] + r(x_k, s_k, d_k, \sigma_k) \ ,$$

with

(3.7) $$\lim_{k\to\infty} \frac{r(x_k, s_k, d_k, \sigma_k)}{\sigma_k^2} = 0 \ .$$

Hence, combining (3.5) and (3.6) gives

$$(3.8) \qquad \frac{-r(x_k, s_k, d_k, \sigma_k)}{\sigma_k^2} \geq -(1-\mu)\left[g_k^t s_k + \frac{1}{2} d_k^t G_k d_k\right] .$$

The conclusion follows from (3.7) and (3.8). □

The result presented by McCormick did not specify a choice of x_{k+1} when x_k was not an indefinite point, but did suggest the Newton direction. In the case that x_k was an indefinite point then $s_k = (\|g_k\|/\|p_k\|)p_k$ with p_k a descent direction such that $-g_k^t p_k \geq c_1\|g_k\|$. Also, d_k was required to be a unit vector such that $d_k^t G_k d_k \leq c_2 \lambda_{G_k}$ where λ_{G_k} is defined as the most negative eigenvalue of G_k. In the above statements $c_1, c_2 > 0$. McCormick was able to conclude that if infinitely many indefinite points $\{x_{k_j}\}$ were to occur in the sequence $\{x_k\}$, then any point of accumulation $\bar{x}$ of the sequence $\{x_{k_j}\}$ must satisfy $g(\bar{x}) = 0$, and $G(\bar{x})$ is positive semidefinite with at least one zero eigenvalue. A specific choice of s_k and d_k was not suggested.

Under the additional hypothesis that the number of critical points in $\mathcal{D}$ is finite, and with a judicious choice of (s_k, d_k) one can show that the iterates defined by (3.1) and (3.2) converge to a point x^* where $g(x^*) = 0$, and $G(x^*)$ is positive semidefinite. However, Armijo type steplength procedures do not take into account any information about the shape of the function along the curve x_α. More sophisticated strategies are available for determining the steplength α_k.

In the rest of this chapter we shall be concerned with the choice of (s_k, d_k), and with a steplength procedure which specifies criteria for terminating a univariate search along curves x_α of the form (2.1). Finally, a convergence result will be given that indicates these choices are quite reasonable.

4. Determining Directions of Negative Curvature

As we shall see, the results of Theorem (3.1) are useful only if

$$(4.1) \qquad (g_k^t s_k \to 0) \Rightarrow (g_k \to 0) ,$$

and

$$(4.2) \qquad (d_k^t G_k d_k \to 0) \Rightarrow (\lambda_{G_k} \to 0) ,$$

where λ_{G_k} is defined to be the most negative eigenvalue of G_k when x_k is an indefinite point and zero otherwise. Intuitively, if (4.1) and (4.2) hold then the iterates $\{x_k\}$ are converging to a critical point where the Hessian is positive semidefinite. These statements will be made precise in sections 5 and 6. Here we present various ways in which (4.2) can be accomplished. Matrix factorizations will play an important role. The factorizations we shall discuss in some detail are Gill and Murray's modified Cholesky factorization [11], and the method of Bunch and Parlett [5].

Gill and Murray present an algorithm which for any symmetric matrix A produces a unit lower triangular matrix L, a diagonal matrix D with positive diagonal elements, and a diagonal matrix E with nonnegative diagonal elements such that

$$A+E = LDL^t .$$

The elements of $LD^{\frac{1}{2}}$ and E are bounded relative to the maximum element of A. This factorization depends upon nonnegative parameters (δ,β). The parameter β is used to force a bound upon the elements of $LD^{\frac{1}{2}}$. The parameter δ in a sense determines the level of positive definiteness that the matrix A+E is required to have. Given the parameter $\delta \geq 0$, this factorization proceeds much the same as the Cholesky factorization

with the exception that when a diagonal element is found to be less than or equal to δ, it is modified. This modification is expressed in the diagonal matrix E.

It is possible to obtain a direction of negative curvature from this factorization when $\delta = 0$. Assuming A has a negative eigenvalue, one computes an index ℓ such that $D_{\ell\ell}-E_{\ell\ell} \leq D_{jj}-E_{jj}$ for $1 \leq j \leq n$. Then the solution d to the equation

$$L^t d = e_\ell \ ,$$

where e_ℓ is the unit vector whose ℓ-th component is 1, can be shown to be a direction of negative curvature.

With this factorization A can have a negative eigenvalue only if E is nonzero. However, when $\delta > 0$ it is possible for E to be nonzero even though A is positive definite. Thus the direction d obtained above cannot be guaranteed to be a direction of negative curvature unless $\delta = 0$. Unfortunately, when this factorization is used in a modified Newton's method $\delta > 0$ must be specified to obtain a proof of convergence.

The factorization of Bunch and Parlett allows an alternative that avoids this difficulty. We have already discussed this factorization in chapters I and II, but we wish to emphasize here the properties of this factorization relevant to this discussion.

Given any symmetric matrix A the factorization will obtain a permutation matrix Q, a block diagonal matrix D, and a unit lower triangular matrix M such that

$$QAQ^t = MDM^t \ .$$

The matrices M and D satisfy

(4.3) The elements of M are bounded by a fixed positive constant which is independent of the matrix A.

(4.4) D is a block diagonal matrix with one-by-one or two-by-two diagonal blocks.

(4.5) D has the same number of positive, negative, and zero eigenvalues as A (Sylvester's Inertia Theorem).

(4.6) The number of 2×2 diagonal blocks plus the number of negative diagonal elements which occur as 1×1 diagonal blocks of D is equal to the number of negative eigenvalues of A. In the case that A is positive semi-definite, D is a diagonal matrix with nonnegative diagonal elements.

The following lemma will show how this factorization can be used to obtain directions of negative curvature which satisfy (4.2).

Lemma (4.1). Let $A = WBW^t$ where $W \in R^{n\times n}$ is nonsingular, and $B \in R^{n\times n}$ is symmetric. Assume that A has at least one negative eigenvalue. Let $\{z_j: j=1,2,\ldots,m\}$ be unit eigenvectors for B corresponding to eigenvalues

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_m < 0 .$$

Let $z = \sum_{j=1}^{k} z_j$ where $1 \leq k \leq m$ and let

(4.7) $$W^t y = z .$$

Then

$$\lambda_A \geq k^2 [K_2(W)]^2 \frac{y^t A y}{y^t y}$$

where λ_A is the smallest eigenvalue of A, and $K_2(W) = \|W\|\|W^{-1}\|$ is the Euclidean condition number of W.

Proof: If x is a unit eigenvector for A corresponding to λ_A then $x^tAx = \lambda_A$ and if $u = W^tx$ then

$$\lambda_A = x^tAx = u^tBu \geq \lambda_1\|u\|^2 .$$

Moreover, since $\|u\| \leq \|W\|$, and $\lambda_1 < 0$,

$$\lambda_A \geq \lambda_1\|W\|^2 . \tag{4.8}$$

Now note that from (4.7)

$$y^tAy = \left(\sum_{j=1}^{k} z_j\right) B \left(\sum_{j=1}^{k} z_j\right) = \sum_{j=1}^{k} \lambda_j < 0 .$$

Since $\|y\| = \|W^{-t}\left(\sum_{j=1}^{k} z_j\right)\| \leq k\|W^{-1}\|$ we have

$$\frac{y^tAy}{y^ty} \leq \frac{\sum_{j=1}^{k} \lambda_j}{k^2\|W^{-1}\|^2} \leq \frac{\lambda_1}{k^2\|W^{-1}\|^2} . \tag{4.9}$$

Together, inequalities (4.8) and (4.9) give the desired result. □

If Lemma (4.1) is to be useful, then $W^ty = z$ must be easy to solve. Also, the eigensystem of B must be readily available, and the factorization $A = WBW^t$ should be relatively cheap to compute. These requirements rule out a full eigensystem decomposition of A and also the factorization of Aasen [1] which gives B in tridiagonal form. However, the Bunch-Parlett factorization certainly satisfies all these requirements with the additional feature that $K_2(W)$ has a bound that is independent of A.

Fletcher and Freeman [8] have suggested the use of this factorization to obtain a direction of negative curvature. The direction they

suggest corresponds to taking k = m in Lemma (4.1). However, Lemma (4.1) suggests that the best direction to use is with k = 1 since this reduces the magnitude of the constant $k^2[K_2(W)]^2$ and is slightly cheaper to compute.

5. A Steplength Algorithm

Once a descent pair (s,d) has been determined at a point x then we are faced with the problem of determining $\bar{\alpha}$ such that

$$f(x_{\bar{\alpha}}) \leq f(x)$$

where $x_\alpha = x+\alpha^2 s+\alpha d$, $0 < \alpha$. One solution would be to determine $\bar{\alpha}$ such that

$$f(x_{\bar{\alpha}}) = \min_{\alpha \geq 0} f(x_\alpha) \ , \tag{5.1}$$

but this is a very difficult computational problem. It is computationally more desirable to replace the problem of satisfying (5.1) exactly with the specification of criteria for terminating a univariate minimization procedure that is designed to solve (5.1).

Such an approach is motivated by the success of previous algorithms which have been used when a single descent direction is specified. Given a descent direction s at a point x, one such algorithm is to terminate the line search when an $\bar{\alpha}$ has been found which satisfies

$$g(x+\bar{\alpha}s)^t s \geq \eta g(x)^t s \ , \tag{5.2}$$

and

$$f(x+\bar{\alpha}s) \leq f(x) + \bar{\alpha}\mu g(x)^t s \ , \tag{5.3}$$

where $0 \leq \mu \leq \eta < 1$ are preassigned constants. If a sequence of points $\{x_k\}$ are determined where $x_{k+1} = x_k+\alpha_k s_k$ with $x = x_k$, $s = s_k$, $\bar{\alpha} = \alpha_k$

satisfying (5.2) and (5.3) for each k, then

$$(g_{k+1}-g_k)^t s_k \geq -(1-\eta) g_k^t s_k ,$$

and hence

$$\|g_{k+1}-g_k\| \geq -(1-\eta) g_k^t s_k / \|s_k\| . \tag{5.4}$$

It follows from (5.4) that

$$\alpha_k \|s_k\| \geq \psi(-(1-\eta) g_k^t s_k / \|s_k\|) \tag{5.5}$$

where ψ is the reverse modulus of continuity of g [17, p. 482]. Since $f_{k+1} \leq f_k$, and f must be bounded below on the compact set $L(x_0)$, we have that $\lim_{k\to\infty} (f_k - f_{k+1}) = 0$ and thus (5.3) implies

$$(\alpha_k \|s_k\|)\ g_k^t s_k / \|s_k\| \to 0 . \tag{5.6}$$

Since $\psi(t_k) \to 0$ implies $t_k \to 0$ it follows from (5.5) and (5.6) that

$$\lim_{k\to\infty} g_k^t s_k / \|s_k\| = 0 . \tag{5.7}$$

Usually g_k and s_k are related so that (5.7) implies $\|g_k\| \to 0$ which in turn implies $\|s_k\| \to 0$. Thus it is concluded that $\|x_{k+1}-x_k\| \to 0$ and $\|g_k\| \to 0$ as long as the α_k are bounded. This is enough to insure that

$$\lim_{k\to\infty} x_k = x^*$$

with x* a critical point of f due to the following lemma given in [17].

Lemma (5.1). Let $f: \mathcal{D} \subset R^n \to R$ be continuously differentiable on the compact set $\mathcal{D}_0 \subset \mathcal{D}$. Let

$$S = \{\hat{x}: \hat{x} \in \mathcal{D}_0,\ g(\hat{x}) = 0\} ,$$

and assume that S is finite. If $\{x_k\} \subset \mathcal{D}_0$ is a sequence such that

$$\lim_{k\to\infty} \|x_{k+1}-x_k\| = 0, \quad \lim_{k\to\infty} \|g_k\| = 0 ,$$

then $\lim_{k\to\infty} x_k = x^*$, where $x^* \in S$.

Proof: See [17, p. 476]. □

A full discussion of this type of strategy may be found in [17]. Particular algorithms of this type are given in [12,17]. The strategy has a geometrical interpretation which is depicted in figure 2.

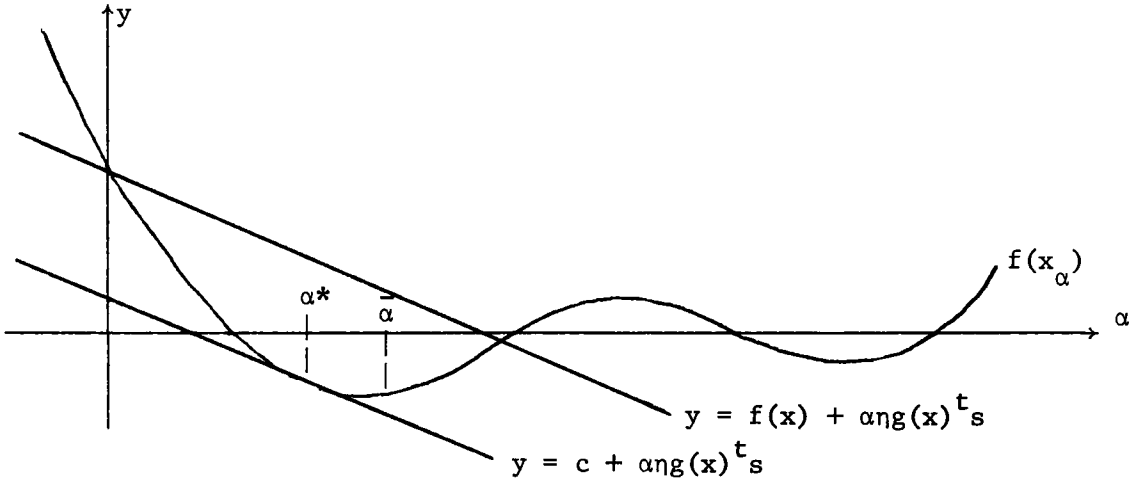


Figure 2
A Search Along x+αs

Here $\mu = \eta$ and α^* is the smallest positive root of the equation $g(x+\alpha s)^t s = \eta g(x)^t s$. The local quadratic approximation to $f(x+\alpha p)$ is

$$\phi(\alpha) = f(x) + \alpha g(x)^t s + \frac{1}{2}\alpha^2 s^t G(x) s$$

which is convex near $\alpha = 0$ if $G(x)$ is positive definite as shown in figure 2. Condition (5.3) guarantees sufficient decrease of the function so that $\|g_k\| \to 0$ which means that $f(x+\alpha p)$ lies below the top line in figure 2. Condition (5.2) guarantees that the distance $\|x_{k+1} - x_k\|$ does not become arbitrarily small. The picture indicates that the only possibility for $\alpha^* \leq \alpha_k$ to be small is that x is close to a local minimum.

The termination criterion we shall give may be viewed as an extension of these ideas which are suitable for the situation when an iterate x_k is an indefinite point. We replace (5.2) and (5.3) with the following rule. If (s,d) is a descent pair at x then we terminate the search when $\bar{\alpha}$ has been found which satisfies

$$g(x_{\bar{\alpha}})^t(2\bar{\alpha}s + d) \geq \eta[g^t s + 2\bar{\alpha}(g^t s + \tfrac{1}{2}d^t G d)] , \tag{5.8}$$

and

$$f(x_{\bar{\alpha}}) \leq f + \mu\bar{\alpha}^2[g^t s + \tfrac{1}{2}d^t G d] , \tag{5.9}$$

with $0 \leq \mu \leq \eta < 1$ as before. Note that when $d = 0$ these conditions reduce to those of (5.2) and (5.3). Again there is a geometrical interpretation which is depicted in figure 3.

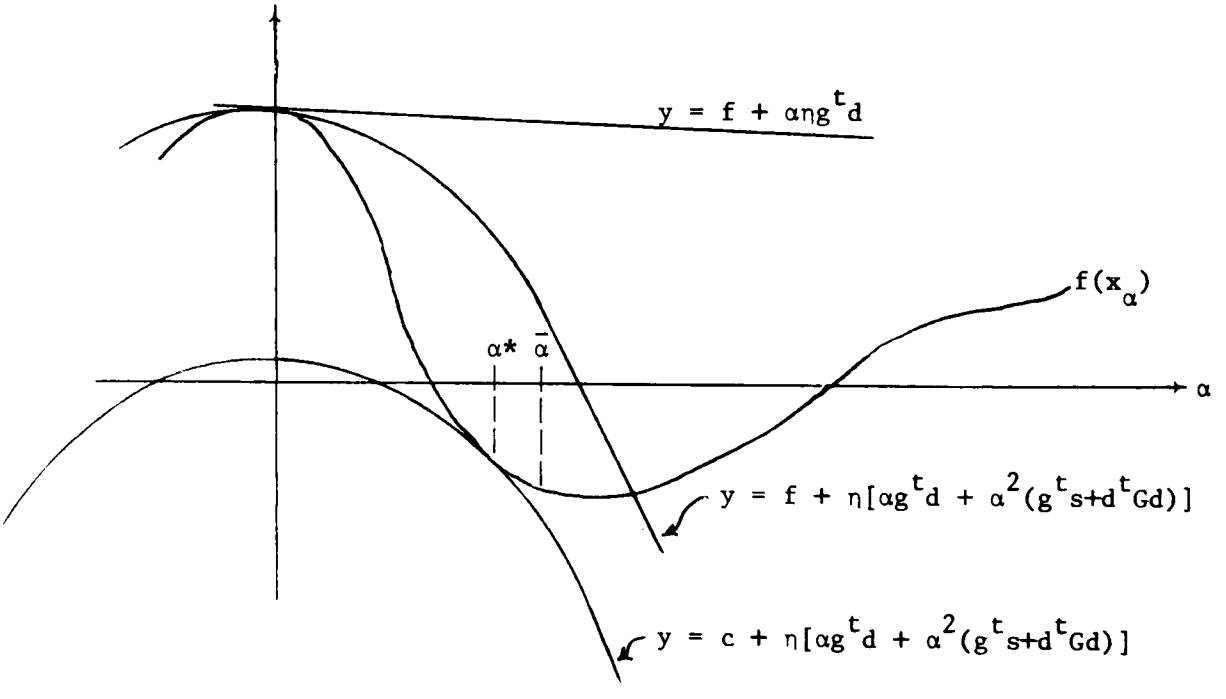


Figure 3
A Search Along $x+\alpha^2 s+\alpha d$

Here α^* is the smallest positive root of the equation

$$g(x_\alpha)^t(2\alpha s+d) = \eta[g^t d + 2\alpha(g^t s + \tfrac{1}{2}d^t G d)] \ .$$

The situation shown in figure 3 describes the shape of $f(x_\alpha)$ along the curve

$$C: \{x_\alpha : x_\alpha = x + \alpha^2 s+\alpha d\} \ ,$$

where x is an indefinite point (see figure 4).

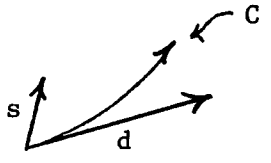


Figure 4
The Curve $x+\alpha^2 s+\alpha d$

An additional requirement is placed on a steplength algorithm at an indefinite point. Sufficient decrease of the function must be used to force the negative eigenvalues of the Hessian to zero as well as to force the gradient to zero. This is guaranteed by condition (5.9). In addition to this we must not let $\|x_{k+1} - x_k\|$ become arbitrarily small. This is accomplished by condition (5.8). The α^* pictured in figure 3 is similar to its counterpart in figure 2. The picture suggests that the only possibility for α^* to become small is for the iterate x_k to be close to a local minimum. The inflection point which must occur along the path C must either be crossed or become "flattened out" in the iterative process.

We note with Fletcher and Freeman [8] that if a direction d_k of negative curvature alone is used (taking $s_k = 0$) then the condition

$|g_{k+1}^t d_k| \leq -\eta g_k^t d_k$ is inappropriate for termination of the linear search because $g_k^t d_k$ may be close to zero even far away from a minimum. They found it necessary to give termination criteria based on an estimate of the first derivative of $f(x_\alpha)$ at the inflection point. The estimate was obtained from the value of the derivative of a related quartic polynomial at its corresponding inflection point.

The following lemma will show that conditions (5.8) and (5.9) can be satisfied whenever a descent pair exists at a point x.

Lemma (5.2). Let $\Phi: R \to R$ be twice continuously differentiable in an open interval I which contains the origin; and suppose that $L(0) \subset I$ is compact where $L(0) = \{\alpha \in I: \Phi(\alpha) \leq \Phi(0)\}$. Let $\mu \in [0,1)$ and $\eta \in [\mu,1)$. Then if $\Phi'(0) \leq 0$ and $\Phi''(0) < 0$ there is an $\bar{\alpha} \in (0,\infty) \cap I$ such that

$$\Phi'(\bar{\alpha}) \geq \eta[\Phi'(0) + \Phi''(0)\bar{\alpha}] \,, \tag{5.10}$$

and

$$\Phi(\bar{\alpha}) \leq \Phi(0) + \mu[\Phi'(0)\bar{\alpha} + \Phi''(0)\frac{\bar{\alpha}^2}{2}] \,. \tag{5.11}$$

Proof: The assumption that $\Phi'(0) \leq 0$ and $\Phi''(0) < 0$ implies the existence of $\hat{\beta} \in I$ with $\Phi(\alpha) < \Phi(0)$ for $0 < \alpha < \hat{\beta}$. Let $\beta = \sup\{\hat{\beta}: \Phi(\alpha) < \Phi(0)$ with $0 < \alpha < \hat{\beta}\}$. Then $\beta > 0$, and the assumption on $L(0)$ implies $\beta \in I$ is finite. The continuity of Φ implies $\Phi(0) = \Phi(\beta)$. Thus

$$\Phi(\beta) \geq \Phi(0) + \mu[\Phi'(0)\beta + \Phi''(0)\frac{\beta^2}{2}] \,, \tag{5.12}$$

Define $h: I \to R$ by

$$h(\alpha) = \Phi(\alpha) - \Phi(0) - \eta[\Phi'(0)\alpha + \Phi''(0)\frac{\alpha^2}{2}] \,.$$

Since $\mu \leq \eta$ we have $h(\beta) \geq 0$. Note also that $h(0) = 0$, $h'(0) \leq 0$, $h''(0) < 0$. This together with the continuity of h implies the existence of $\beta_1 \in (0,\beta]$ such that $h(\beta_1) = 0$, and $h(\alpha) < 0$ for all $\alpha \in (0,\beta_1)$. Now

Rolle's Theorem implies the existence of $\bar{\alpha} \in (0,\beta_1)$ such that $h'(\bar{\alpha}) = 0$, and (5.10) follows. Also, $h(\bar{\alpha}) < 0$ and $\mu \leq \eta$ imply (5.11). □

If we take $\Phi(\alpha) = f(x_\alpha)$ then Lemma (5.2) implies that conditions (5.8) and (5.9) can be satisfied. In the next section we will show how these conditions may be used to prove the convergence of a modified Newton method.

6. Convergence of the Modified Newton Iteration

Now we turn our attention to defining a modified Newton iteration. We shall give a convergence result based on the use of descent pairs and the steplength algorithm discussed above. The proof proceeds in two parts. The first result is somewhat independent of the definition of the iterates. The second part will use the particular way in which the iterates are defined to establish convergence.

The general iteration from a point x_k begins with determining a descent pair (s_k,d_k) at x_k. Let

$$\Phi_k(\alpha) = f(x_k + \alpha^2 s_k + \alpha d_k) \quad . \tag{6.1}$$

Assume $\mu \in (0,1)$ and $\eta \in [\mu,1)$ are independent of k. Then $\alpha_k > 0$ is determined such that

$$y_k = x_k + \alpha_k^2 s_k + \alpha_k d_k \in \mathcal{D} \quad , \tag{6.2}$$

$$f(y_k) \leq f(x_k) + \mu \Phi_k''(0) \frac{\alpha_k^2}{2} \quad , \tag{6.3}$$

$$\Phi_k'(\alpha_k) \geq \eta[\Phi_k'(0) + \Phi_k''(0)\alpha_k] \quad . \tag{6.4}$$

Take $x_{k+1} = y_k$.

One might note that due to (5.11) in the statement of Lemma (5.2)

we could require $f(y_k) \leq f(x_k) + \mu\left[\Phi_k'(0)\alpha_k + \Phi_k''(0)\frac{\alpha_k^2}{2}\right]$ instead of (6.3). However, the additional term does not enhance the convergence result in any way, while it does give a more stringent requirement to be satisfied by the univariate search. The first step in the convergence result is

Theorem (6.1). Let f satisfy assumptions (1.2). Then the iteration defined above satisfies

$$\lim_{k\to\infty} -g_k^t s_k = 0 , \tag{6.5}$$

and

$$\lim_{k\to\infty} -d_k^t G_k d_k = 0 . \tag{6.6}$$

Proof: From (2.7) and (2.8) we have $\Phi_k'(0) = g_k^t d_k$ and $\Phi_k''(0) = 2g_k^t s_k + d_k^t G_k d_k$. Since (s_k, d_k) is a descent pair, $\Phi_k'(0) \leq 0$, and $\Phi_k''(0) < 0$. Thus (6.3) implies that $\{x_k\} \subset L(x_0)$. By the continuity of f and compactness of $L(x_0)$ we have $\lim_{k\to\infty}(f_k - f_{k+1}) = 0$. Now

$f_k - f_{k+1} \geq -\mu\Phi_k''(0)\frac{\alpha_k^2}{2} \geq 0$, so that

$$\lim_{k\to\infty} -\alpha_k^2 g_k^t s_k = 0 , \tag{6.7}$$

and

$$\lim_{k\to\infty} -\alpha_k^2 d_k^t G_k d_k = 0 . \tag{6.8}$$

From condition (6.4) we obtain

$$\Phi_k'(\alpha_k) - \Phi_k'(0) - \alpha_k\Phi_k''(0) \geq -(1-\eta)[\Phi_k'(0) + \Phi_k''(0)\alpha_k] ,$$

and hence

$$\Phi_k'(\alpha_k) - \Phi_k'(0) - \alpha_k\Phi_k''(0) \geq -(1-\eta)\Phi_k''(0)\alpha_k ,$$

An application of the mean value theorem now yields that for some $\theta_k \in (0,\alpha_k)$,

(6.9) $$\Phi_k''(\theta_k) - \Phi_k''(0) \geq -(1-\eta)\Phi_k''(0) .$$

The desired result now follows readily, for if either (6.5) or (6.6) do not hold, then there is a subsequence $\{k_i\}$ and a $\sigma > 0$ such that

(6.10) $$-\Phi_{k_i}''(0) \geq \sigma > 0 .$$

Hence (6.9) implies that $\{\alpha_{k_i}\}$ does not converge to zero. However, if $\{\alpha_{k_i}\}$ does not converge to zero and (6.10) holds, then (6.7) and (6.8) cannot be satisfied. This contradiction establishes the theorem. □

The $\{\alpha_k\}$ of (6.2)-(6.4) are to be determined by a univariate minimization algorithm applied to $\Phi_k(\alpha)$. Let $\beta > 0$ be fixed, and terminate the search when $0 < \bar{\alpha} \leq \beta$ has been found such that (6.4) is satisfied with $\bar{\alpha}$ in place of α_k. If (6.3) is also satisfied we accept $\alpha_k = \bar{\alpha}$. If either (6.4) cannot be satisfied (say within a fixed number of steps) or if $\bar{\alpha}$ does not satisfy (6.3) we take ω to be the largest element of the set $\{2^{-i}: i=0,1,2,\ldots\}$ such that (6.3) is satisfied with $\bar{\alpha}\omega$ in place of α_k and then accept $\alpha_k = \bar{\alpha}\omega$. If infinitely many of the α_k's must be determined in this way, then Theorem (3.1) applies so that (6.5) and (6.6) are still obtained. We shall call this process the steplength rule $SR(\mu,\eta,\beta)$.

Our next result will show that the iterates defined by this steplength rule converge to a critical point of f where the Hessian is positive semidefinite. It is here that specific properties of the descent pairs (s_k,d_k) are crucial.

Theorem (6.2). Assume in addition to the hypothesis of Theorem (6.1) that f has finitely many critical points in $L(x_0)$. Suppose that the sequence $\{x_k: k=0,1,2,\ldots\}$ has been obtained using the steplength rule

$SR(\mu,\eta,\beta)$ where the descent pairs (s_k,d_k) satisfy $\{\|s_k\|, \|d_k\|: k=0,1,2,\ldots\}$ is bounded together with

$$(6.12) \qquad (g_k^t s_k \to 0) \Rightarrow (g_k \to 0 \text{ and } s_k \to 0) ,$$

and

$$(6.13) \qquad (d_k^t G_k d_k \to 0) \Rightarrow (\lambda_{G_k} \to 0 \text{ and } d_k \to 0) ,$$

as $k \to \infty$. Then

$$\lim_{k\to\infty} x_k = x^*$$

with $g(x^*) = 0$ and $G(x^*)$ positive semidefinite. Moreover, if infinitely many of the x_k are indefinite points, then $G(x^*)$ must have at least one zero eigenvalue.

Proof: From Theorem (6.1) we see that $\lim g_k^t s_k = 0$ and $\lim d_k^t G_k d_k = 0$. By (6.12) we have $g_k \to 0$ and $s_k \to 0$. By (6.13) we have $\lambda_{G_k} \to 0$ and $d_k \to 0$. Now,

$$\|x_{k+1} - x_k\| \leq \beta^2\|s_k\| + \beta\|d_k\| ,$$

hence

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0 .$$

Therefore, Lemma (5.1) applies and we obtain

$$\lim_{k\to\infty} x_k = x^* ,$$

with $g(x^*) = 0$. Since $\lambda_{G_k} \to 0$, we also have by the continuity of G that $G(x^*)$ must be positive semidefinite. Moreover, if infinitely many of the x_k are indefinite points then every neighborhood of x^* contains an indefinite point. Thus the continuity of G implies that $G(x^*)$ has at least one zero eigenvalue.

□

Obviously, the proof of Theorem (6.1) rests on the steplength

rule, while the proof of Theorem (6.2) rests on the particular choice of the descent pairs. Many choices of s_k are possible which satisfy (6.12). Indeed, if A_k is any sequence of symmetric positive definite matrices such that $\|A_k\|, \|A_k^{-1}\|$ are bounded independently of k, then choosing s_k as the solution of

$$A_k s_k = -g_k$$

will satisfy (6.12).

In section 3 we gave several ways to choose the d_k at an indefinite point so that

$$(d_k^t G_k d_k \to 0) \Rightarrow (\lambda_{G_k} \to 0) .$$

The additional requirements that d_k must satisfy are obtained if we replace d_k with $\pm\gamma(\lambda_{G_k})d_k$, where γ is a function such that $\gamma(t_k) \to 0 \Rightarrow t_k \to 0$, and where the sign is chosen to make $g_k^t d_k \leq 0$.

The iterates should also reduce naturally to Newton's iteration as soon as a region is found where the Hessian is positive definite. Indeed, the main motivation for this strategy is to obtain the iterates using second derivative information which is based on the <u>true quadratic model</u> at each x_k. Of course, it is expected that in practice very few indefinite points will be encountered during the iterative process. In fact, Theorem (6.2) indicates that the strategy we have presented <u>actively</u> seeks a region where the Hessian matrix is positive semi-definite. If, for example, the Hessian G(x) is nonsingular whenever x is a critical point of f then only finitely many of the iterates can be indefinite points.

Finally, we shall suggest a way to obtain the descent pairs

(s_k,d_k) which satisfy all of the requirements of Theorem (6.2). In our description we assume $G_k = M_k D_k M_k^t$ is the Bunch-Parlett factorization of the Hessian. Thus we have omitted explicit representation of the permutations Q_k which will be present in practice. Given f and x_0 which satisfy the hypothesis of Theorem (6.2), for k=0,1,2,... define

(6.14) s_k as the solution of

$$(M_k \bar{D}_k M_k^t) s_k = -g_k$$

where $\bar{D}_k = U_k \bar{\Lambda}_k U_k^t$ is obtained from D_k by first obtaining the eigensystem $D_k = U_k \Lambda_k U_k^t$ of D_k and then replacing the diagonal elements $\lambda_j^{(k)}$ of Λ_k with

$$\max_{1\le j\le n} (|\lambda_j^{(k)}|, \varepsilon n \max_{1\le i\le n} |\lambda_i^{(k)}|, \varepsilon) ,$$

where ε is the relative machine precision. In the decomposition of D_k we have $U_k^t U_k = I$, and Λ_k diagonal. Note that only O(n) arithmetic operations are required to obtain $\bar{D}_k$ from D_k.

(6.15) d_k is the solution to

$$M_k^t d_k = \pm |\lambda_{D_k}|^{\frac{1}{2}} z_k$$

where λ_{D_k} is the most negative eigenvalue and z_k the corresponding unit eigenvector of D_k. When D_k does not have a negative eigenvalue we take $d_k = 0$.

The compactness of $L(x_0)$ and the continuity of G imply that the elements of G_k and the components of g_k are uniformly bounded. Thus (s_k,d_k) satisfy the requirements of a descent pair as well as (6.12) and (6.13) due to the bound on the condition numbers $K_2(M_k)$.

The above choice of (s_k,d_k) is somewhat ad hoc and we make no mathematical statements concerning the desirability of this choice. However, in the next chapter computational results will be reported which show that this specification of (s_k,d_k) works reasonably well in practice. We wish to emphasize that many other choices are possible.

We have not addressed the problem of providing an initial step α to the univariate search. Many strategies for determining the initial step are possible. However, we have not found a strategy with enough theoretical basis to recommend it over something very simple such as taking the initial step to be $\alpha = 1$ each time. Note, however, that whatever strategy is chosen must eventually take $\alpha = 1$ in order to retain the local quadratic rate of convergence enjoyed by Newton's method.

7. Conclusions

The algorithm we have just described has the following informal description:

(7.1)

Given $x_0 \in \mathcal{D}$

for k=0,1,2,...

(1) Determine a descent pair (s_k,d_k)

(2) Determine α_k by SR(μ,η,β)

(3) $x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k$.

Step (1) involves evaluating and factoring the Hessian G_k. Step (2) involves the use of a univariate search that can satisfy SR(μ,η,β).

The importance of this iteration is that it represents a natural extension of previous theory to include second derivative information. It avoids saddle points and possesses a strong theoretical convergence

property. Finally, the iteration, even in this preliminary stage of development, performs well in practice.

Chapter V

Computational Results

1. Introduction

The purpose of this chapter is to present computational support of the theoretical results obtained in chapters II, III, and IV. The updating algorithm was tested for timing and accuracy on a large number of random updating problems. The optimization algorithm was tested on a set of test problems which have been used extensively at Argonne National Laboratory for such purposes [5]. In addition to this, the algorithm was tested on some problems which demonstrate its behavior when many indefinite points are encountered during an iteration.

2. Testing the Updating Algorithm

There are two important criteria for testing an updating algorithm. The first criterion is that the updating algorithm actually should represent a computational savings over the alternative of forming the updated matrix and refactoring. The second criterion is that solutions of linear equations using the updating method should be reasonably close to solutions obtained by forming the updated matrix and refactoring.

Timing the updating algorithm and comparing to the alternative is a straightforward task. In order to address the question of accuracy one must decide what quantities should be measured and compared. For each update it seems reasonable to compare

(2.1) $$\|Ax_c - b\| / \|b\|$$

with

$$\|Ax_u - b\| / \|b\| \tag{2.2}$$

for several right hand sides b. In (2.1) the vector x_c is the solution obtained by forming and refactoring the updated matrix. In (2.2) the vector x_u is the solution obtained by using the updating algorithm. The quantity

$$\|x_c - x_u\| / \|x_c\| \tag{2.3}$$

should also be computed.

The quantities in (2.1) and (2.2) measure the relative error in the residual. This relative residual indicates how close the computed solution is to satisfying the equation Ax = b relative to the size of the right hand side b. The quantity (2.3) measures how much the answer obtained by the updating method has deviated from the answer obtained by computing and refactoring the updated matrix.

The process used to test these criteria can most easily be described by means of an informal algorithm. Given a dimension n, we start with $A = I_n$ the $n \times n$ identity matrix. Then the following iteration is carried out.

(2.4) $A := I$

for k=0,1,2,...,m

(1) $z \in R^n$ is chosen with random components in (-1,1);

(2) $\sigma \in R$ is a random number in (-100,100).

(3) $A = A + \sigma z z^t$

(3.1) $\tilde{Q}_u \tilde{\tilde{A}} \tilde{Q}_u^t = \tilde{M}_u \tilde{D}_u \tilde{M}_u^t$ by updating;

(3.2) $\tilde{Q}_c \tilde{\tilde{A}} \tilde{Q}_c^t = \tilde{M}_c \tilde{D}_c \tilde{M}_c^t$ by forming $\tilde{A}$ and factoring;

(4) for j=1,...,5

(4.1) $b \in R^n$ is chosen with random components in (-50,50);

(4.2) Solve Ax = b

(i) Using (3.1) to compute x_u;

(ii) Using (3.2) to compute x_c;

(4.3) Compute

(i) $\|Ax_c-b\| / \|b\|$;

(ii) $\|Ax_u-b\| / \|b\|$;

(iii) $\|x_c-x_u\| / \|b\|$;

The steps (3.1) and (3.2) of iteration (2.4) were timed. These timings were averaged over the number m of updates. Thus the time required by the updating algorithm can be compared to the time required by the alternative of computing $\tilde{A}$ and refactoring. The solution to $\tilde{A}x = b$ was computed for five different right hand sides after each update. This was done to increase the chances of obtaining a large residual $\|Ax_u-b\| / \|b\|$. The quantities (2.1), (2.2), and (2.3) were averaged over all iterations and right hand sides. The results are shown in tables 2 and 3.

Table 2 shows the above quantities for various values of the dimension n. In Table 2 UAVE is the average value of $\|Ax_u-b\| / \|b\|$, CAVE is the average value of $\|Ax_c-b\| / \|b\|$, and AVERR is the average value of $\|x_c-x_u\| / \|x_c\|$. The quantity CTIME is the average time to compute and refactor $\tilde{A}$ and UTIME is the average time to update the factorization.

Table 2

Results for Increasing Order

n	UAVE	CAVE	AVERR	UTIME	CTIME
5	6×10^{-14}	4×10^{-15}	4×10^{-14}	167	424
10	2×10^{-13}	2×10^{-14}	3×10^{-13}	320	1567
20	1×10^{-13}	3×10^{-14}	1×10^{-13}	706	6459
30	3×10^{-13}	7×10^{-14}	2×10^{-13}	1162	16606
40	8×10^{-13}	2×10^{-13}	4×10^{-13}	1819	32468
50	2×10^{-12}	3×10^{-13}	4×10^{-13}	2533	55016

The times shown here are in microseconds. The important thing to note is the relationship of UTIME to CTIME as n increases. To see that the numbers are in the correct proportion one should compare n^2 to UTIME and $\frac{n^3}{6}$ to CTIME. Observe also that there is roughly only a one digit loss of accuracy using the updating algorithm. For each of the results in Table 2 we have taken m = 100 in (2.4).

Table 3 shows the results of a particular updating sequence computed by the iteration (2.4). In this example n = 10. The updating process was carried out for 1000 updates. The results show every fifth update selected from the beginning, middle, and end of these computations. In Table 3 the quantities are not averaged. UERR is $\|Ax_u - b\| / \|b\|$, CERR is $\|Ax_c - b\| / \|b\|$, and XERR is $\|x_c - x_u\| / \|x_c\|$ for only one right hand side. UTIME and CTIME are the timings for each individual update in this case. For the entire sequence, the average quantities were UAVE = 2×10^{-13}, CAVE = 3×10^{-15}, and AVERR = 1×10^{-13}.

Table 3

Results of a Long Range of Updates

	UERR	CERR	XERR	UTIME	CTIME
beginning	6×10^{-15}	4×10^{-15}	8×10^{-16}	312	1563
	7×10^{-14}	3×10^{-14}	6×10^{-15}	313	1875
	3×10^{-15}	7×10^{-16}	3×10^{-15}	521	1979
	1×10^{-14}	5×10^{-15}	2×10^{-15}	417	2083
	1×10^{-14}	4×10^{-15}	2×10^{-14}	417	1563
middle	5×10^{-14}	6×10^{-16}	5×10^{-14}	312	1667
	2×10^{-13}	1×10^{-15}	7×10^{-14}	313	1562
	8×10^{-14}	6×10^{-16}	4×10^{-14}	208	1563
	7×10^{-14}	1×10^{-15}	5×10^{-14}	312	1980
	1×10^{-13}	4×10^{-16}	1×10^{-13}	312	1667
end	7×10^{-13}	3×10^{-15}	2×10^{-12}	208	1458
	5×10^{-13}	3×10^{-15}	5×10^{-13}	312	1667
	5×10^{-13}	1×10^{-15}	2×10^{-12}	209	1458
	2×10^{-13}	1×10^{-15}	5×10^{-13}	209	1771
	1×10^{-13}	1×10^{-15}	2×10^{-13}	312	1563

These results indicate that the error analysis in Chapter III is somewhat pessimistic. In particular, Table 3 shows that obtaining the factorization by the updating method does not deteriorate much even over a long range of updates. The timings show that the operation count given in Chapter II was indeed a worst case analysis. They indicate that the worst case rarely happens. This is demonstrated in Table 3 since for matrices of order 10 the operation count predicts that the updating algorithm should require as much work as the alternative.

One disadvantage of the updating algorithm is the length of computer code necessary to describe the algorithm. The timing results indicate that it would be a worthwhile project to see if the length of code could be decreased; perhaps at the expense of increasing the operation count slightly.

3. Testing the Modified Newton's Method

The unconstrained optimization algorithm described in Chapter IV was tested on some standard minimization problems. The computer implementation is still under development. Therefore, the results presented here are to be regarded as an indication that the method is promising. There are a number of practical considerations that must be settled before this algorithm can be recommended for general use.

One of the practical problems is the choosing of the descent direction s at an indefinite point. We have described one way in Chapter IV, but we feel that others should be tried. Also, it is not clear what the scaling of the descent direction s should be relative to the direction of negative curvature d.

Another problem is choosing the initial step for the linear search procedure at an indefinite point. Enough information is available at an indefinite point to use a cubic polynomial to predict an initial step. To do this, one interpolates f, f', f'' at x where the derivatives are taken along the curve $x+\alpha^2 s+\alpha d$. The resulting cubic polynomial is then required to achieve a decrease Δ at its local minimum $\hat{\alpha}$. The number Δ is the amount of decrease obtained on the last iteration. This process uniquely defines a polynomial p. We then have

$$\hat{\alpha} = (3\Delta - 2f')/f'' \ .$$

We also require that the initial step α_0 satisfy $.5 \leq \alpha_0 \leq 1$. Thus we take $\alpha_0 = \hat{\alpha}$ if $.5 \leq \hat{\alpha} \leq 1$. Otherwise we take the closest endpoint to $\hat{\alpha}$. Obviously there is little theoretical justification for this choice of α_0, but it does an adequate job when safeguarded as mentioned.

Finally, there is always the task of choosing parameters. For instance we must specify μ, η, and β for the steplength rule SR(μ,η,β) (see Chapter IV). In addition to this we must specify criteria for accepting an iterate as an approximation to a local minimum. This, of course, requires the specification of other parameters.

In the following examples we have taken $\mu = 10^{-4}$, $\eta = .9$, and $\beta = 10^{6}$. An iterate x_k is accepted as an approximation to a local minimum when

(i) The Hessian is positive semidefinite,

(ii) $|f_k - f_{k-1}| < (\tau^2+\varepsilon)(1+|f_k|)$,

(iii) $\alpha_{k-1}\|s_{k-1}\| < (\tau+\sqrt{\varepsilon})(1+\|x_k\|)$,

(iv) $g_k^t g_k < \varepsilon^{2/3}(1+|f_k|)^2$.

Here ε is the relative machine precision. The parameter τ is specified by the user but defaults to $10\sqrt{\varepsilon}$ if found to be smaller than ε. For these examples τ is given the default value. These stopping criteria are used in the Gill and Murray algorithm. We have adopted them in order to obtain a good comparison of the two algorithms. These functions were used as test problems:

(3.1) Rosenbrock's Problem;

$n = 2,$

$f = (1-x_1)^2 + 100(x_2-x_1^2)^2,$

standard start: (-1.2,1.0).

(3.2) Powell's Function of Four Variables;

$n = 4,$

$f = (x_1+10x_2)^2 + 5(x_3-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4,$

standard start: (3.0, -1.0, 0.0, 1.0).

(3.3) Brown's Function with Two Global Minima;

$n = 2,$

$f = (x_1^2-x_2-1)^2 + ((x_1-x_2)^2 + (x_2-0.5)^2 - 1)^2,$

standard start: (0.1, 2.0).

(3.4) Powell's Badly Scaled Function of Two Variables;

$n = 2,$

$f = (10^4 x_1 x_2-1)^2 + (e^{-x_1} + e^{-x_2} - 1.0001)^2,$

standard start: (0.0, 1.0).

(3.5) Box's Function;

$n = 3,$

$f = \sum_{i=1}^{10} \left(e^{-x_1\delta i} - e^{-x_2\delta i} - x_3(e^{-\delta i} - e^{-10\delta i})\right)^2$

where $\delta_i = i/10,$

standard start: (0.0, 20.0, 20.0).

(3.6) Wood's Function;

$n = 4$,

$$f = 100(x_2-x_1^2)^2 + (1-x_1)^2 + 90(x_4-x_3^2)^2 + (1-x_3)^2 + 10.1((x_2-x_1)^2 + (x_4-1)^2) + 19.8(x_2-1)(x_4-1),$$

standard start: (-3.0, -1.0, -3.0, -1.0).

(3.7) Penalty Function I;

$n = 4$,

$$f = A\sum_{i=1}^{n}(x_i-1)^2 + B\left[\sum_{i=1}^{n} x_i^2 - \frac{1}{4}\right]^2, \text{ where } A = 10^{-5},\ B = 1,$$

standard start: $x_i = i$, (for $1 \leq i \leq n$).

(3.8) EXP6;

$n = 6$,

$$f = \sum_{i=1}^{13}\left(x_3e^{-x_1z_i} - x_4e^{-x_2z_i} + x_6e^{-x_5z_i} - y_i\right)^2,$$

where $y_i = e^{-z_i} - 5e^{-10z_i} + 3e^{-4z_i}$,

$z_i = (0.1)i$, (for $1 \leq i \leq 13$),

standard start: (1.0, 2.0, 1.0, 1.0, 1.0, 1.0).

(3.9) Brown's Badly Scaled Problem;

$n = 2$,

$$f = (x_1-10^6)^2 + (x_2-2\times10^{-6})^2 + (x_1x_2-2)^2,$$

standard start: (1.0, 1.0).

(3.10) Beal's Function;

$n = 2$,

$$f = \sum_{i=1}^{3}(c_i-x_1(1-x_2^i))^2,$$

where $c_1 = 1.5$, $c_2 = 2.25$, $c_3 = 2.625$,

standard start: (1.0, 1.0).

(3.11) Rosenbrock's Cliff Function;

$n = 2,$

$$f = \left(\frac{x_1-3}{100}\right)^2 - (x_1-x_2) + e^{20(x_1-x_2)},$$

standard start: (0.0, -1.0).

(3.12) Cubic Function;

$n = 3,$

$$f = \sum_{i=1}^{7} f_i^2 + 2,$$

where $f_1 = x_1^4$, $f_2 = f_3 = 0.1x_1^2(x_2-1)^2$, $f_4 = (x_2-1)^4$,

$f_5 = f_6 = 0.1x_1^2(x_3-1)^2$, $f_7 = (x_3-1)^4$,

standard start: (2.0, -3.0, 3.0).

(3.13) Gottfried's Function;

$n = 2,$

$$f = (x_1 - 0.1136(x_1+3x_2)(1-x_1))^2 + (x_2 + 7.5(2x_1-x_2)(1-x_2))^2,$$

standard start: (0.5, 0.5).

(3.14) Four Cluster Function;

$n = 2,$

$$f = [(x_1-x_2^2)(x_1-\sin(x_2))]^2 + [(\cos(x_2)-x_1)(x_2-\cos(x_1))]^2,$$

standard start: (0, 0).

(3.15) Hyperbola-Circle Function;

$n = 2,$

$$f = (x_1x_2-1)^2 + (x_1^2+x_2^2-4)^2,$$

standard start: (0.0, 1.0).

Table 4 shows the results of these tests on problems (3.1) - (3.15) with the starting point x_0 taken to be the standard start. The results of the Gill and Murray algorithm on the same problems are also given in this table. For each problem the first entry is the result of the algorithm presented in Chapter IV and the second entry is the result of the Gill and Murray algorithm. The quantities represented are:

NITER = the number of Hessian evaluations,

NFEV = the number of function evaluations,

$g^t g = \|g\|^2$,

POSDEF = T if the Hessian was found to be positive semidefinite at the solution, and F otherwise,

NEGCNT = the number of indefinite points encountered during the iteration,

FLAG = 0 means normal termination.

1 means abnormal termination.

(We note that for either algorithm an abnormal termination may have been indicated even though the approximation was close to the solution.)

Table 4

Results of Tests with Standard Starts

#	NITER	NFEV	$g^t g$	POSDEF	NEGCNT	FLAG
3.1	21	28	5×10^{-19}	T	0	0
	23	29	6×10^{-27}			0
3.2	29	30	5×10^{-26}	T	0	0
	25	25	5×10^{-21}			0
3.3	8	10	0.0	T	0	0
	9	10	1×10^{-18}			0
3.4	138	239	1×10^{-9}	T	2	0
	186	344	2×10^{-15}			1
3.5	14	18	2×10^{-27}	T	2	0
	10	10	4×10^{-26}			1
3.6	38	48	2×10^{-17}	T	1	0
	39	50	1×10^{-17}			1
3.7	34	43	2×10^{-23}	T	0	0
	36	44	1×10^{-30}			0
3.8	527	1000	6×10^{-6}	F	527	1
	47	382	4×10^{-24}			0
3.9	8	10	2×10^{-43}	T	1	0
	8	10	2×10^{-43}			0
3.10	9	11	9×10^{-20}	T	2	0
	9	71	2×10^{-21}			0
3.11	27	28	3×10^{-20}	T	0	0
	28	29	1×10^{-20}			1
3.12	66	67	2×10^{-52}	T	0	0
	33	33	1×10^{-24}			0
3.13	8	16	9×10^{-25}	T	3	0
	4	1354	4×10^{-5}			1
3.14	11	12	1×10^{-25}	T	0	0
	14	33	1×10^{-29}			0
3.15	6	7	0.0	T	1	0
	7	8	9×10^{-30}			0

The algorithm presented in Chapter IV requires the calculation of the Hessian from an analytic expression in order for the underlying theory to be valid. However, one may want to use the algorithm with a finite difference approximation to the Hessian. In Table 5 the results of using such an approximation on problems (3.1) - (3.15) are presented. The headings in this table are as in Table 4. Again we use the standard starts for x_0. It should be noted that except for Powell's Badly Scaled Function (3.4), there is little difference between the behavior of the algorithm with finite differences and with analytic derivatives.

Table 5

Results from Using Finite Differences

#	NITER	NFEV	$g^t g$	POSDEF	NEGCNT	FLAG
3.1	21	28	1×10^{-18}	T	0	0
3.2	29	30	7×10^{-26}	T	0	0
3.3	8	10	3×10^{-28}	T	0	0
3.4	553	1000	1×10^{3}	F	546	1
3.5	14	18	3×10^{-25}	T	2	0
3.6	38	48	5×10^{-17}	T	1	0
3.7	34	43	3×10^{-21}	T	0	0
3.8	561	1000	8×10^{-6}	F	561	1
3.9	11	23	2×10^{-19}	T	2	0
3.10	9	11	3×10^{-19}	T	2	0
3.11	34	44	2×10^{-16}	T	16	0
3.12	66	67	2×10^{-52}	T	0	0
3.13	8	16	3×10^{-17}	T	3	0
3.14	11	12	2×10^{-20}	T	1	0
3.15	6	7	6×10^{-30}	T	1	0

The use of standard starting points on these test examples does not fully reveal the performance of this algorithm. Some of the

standard starts are in regions such that little or no negative curvature is encountered during the iteration. In order to demonstrate how the algorithm performs when many indefinite points are encountered, we include results of the algorithm on problems (3.5), (3.8), (3.9), and (3.13) with random starting points. These results are presented in tables 6, 7, 8 and 9. In each table the results from ten random starting points are given. For each point there are two entries. The first is from the algorithm presented in Chapter IV and the second is the result from Gill and Murray's algorithm on the same problem.

Table 6

Box's Function

#	NITER	NFEV	$g^t g$	POSDEF	NEGCNT	FLAG
1	25	36	1×10^{-25}	T	21	0
	24	140	4×10^{-8}			1
2	16	17	2×10^{-31}	T	3	0
	36	97	1×10^{-24}			1
3	14	15	3×10^{-31}	T	3	0
	27	70	5×10^{-27}			1
4	20	26	1×10^{-20}	T	2	0
	20	20	6×10^{-33}			0
5	26	42	0.0	T	22	0
	37	118	5×10^{-25}			1
6	22	41	1×10^{-32}	T	17	0
	26	64	1×10^{-22}			1
7	20	33	1×10^{-22}	T	18	0
	19	26	1×10^{-27}			0
8	18	22	4×10^{-28}	T	3	0
	9	9	2×10^{-21}			0
9	16	20	5×10^{-25}	T	1	0
	14	14	1×10^{-28}			0
10	12	16	8×10^{-32}	T	7	0
	33	95	2×10^{-25}			1

Table 7

EXP6

#	NITER	NFEV	$g^t g$	POSDEF	NEGCNT	FLAG
1	485	730	2×10^{-21}	T	41	0
	2	81	overflow			1
2	6	6	6×10^{7}	F	6	1
	135	198	4×10^{-6}			1
3	25	25	2×10^{8}	F	25	1
	79	147	2×10^{-23}			0
4	202	276	5×10^{-28}	T	200	0
	108	353	2×10			1
5	8	71	6×10^{2}	F	8	1
	98	418	3			1
6	40	431	4×10^{-20}	T	38	0
	128	253	4×10^{-6}			1
7	335	543	2×10^{-19}	T	59	0
	129	227	9×10^{-4}			1
8	98	138	5×10^{-19}	T	32	0
	88	474	4			1
9	865	1000	2×10^{-4}	F	865	1
	112	332	2×10^{-1}			1
10	810	1000	5×10^{-5}	F	810	1
	108	358	1×10^{-8}			1

Table 8

Gottfried's Function

#	NITER	NFEV	$g^t g$	POSDEF	NEGCNT	FLAG
1	26	29	2×10^{-19}	T	3	0
	27	29	2×10^{-30}			0
2	18	26	3×10^{-18}	T	3	0
	19	22	5×10^{-29}			
3	12	17	8×10^{-29}	T	3	0
	14	15	1×10^{-22}			0
4	21	29	1×10^{-20}	T	6	0
	1	1001	2×10^{13}			1
5	19	27	5×10^{-20}	T	8	0
	1	1001	2×10^{15}			1
6	13	15	1×10^{-27}	T	3	0
	1	1001	9×10^{12}			1
7	25	27	1×10^{-19}	T	2	0
	25	26	4×10^{-22}			0
8	21	25	8×10^{-21}	T	4	0
	1	1001	2×10^{15}			1
9	15	16	1×10^{-27}	T	2	0
	1	1001	4×10^{16}			1
10	14	16	1×10^{-26}	T	3	0
	1	1001	3×10^{14}			1

Table 9

Brown's Badly Scaled Problem

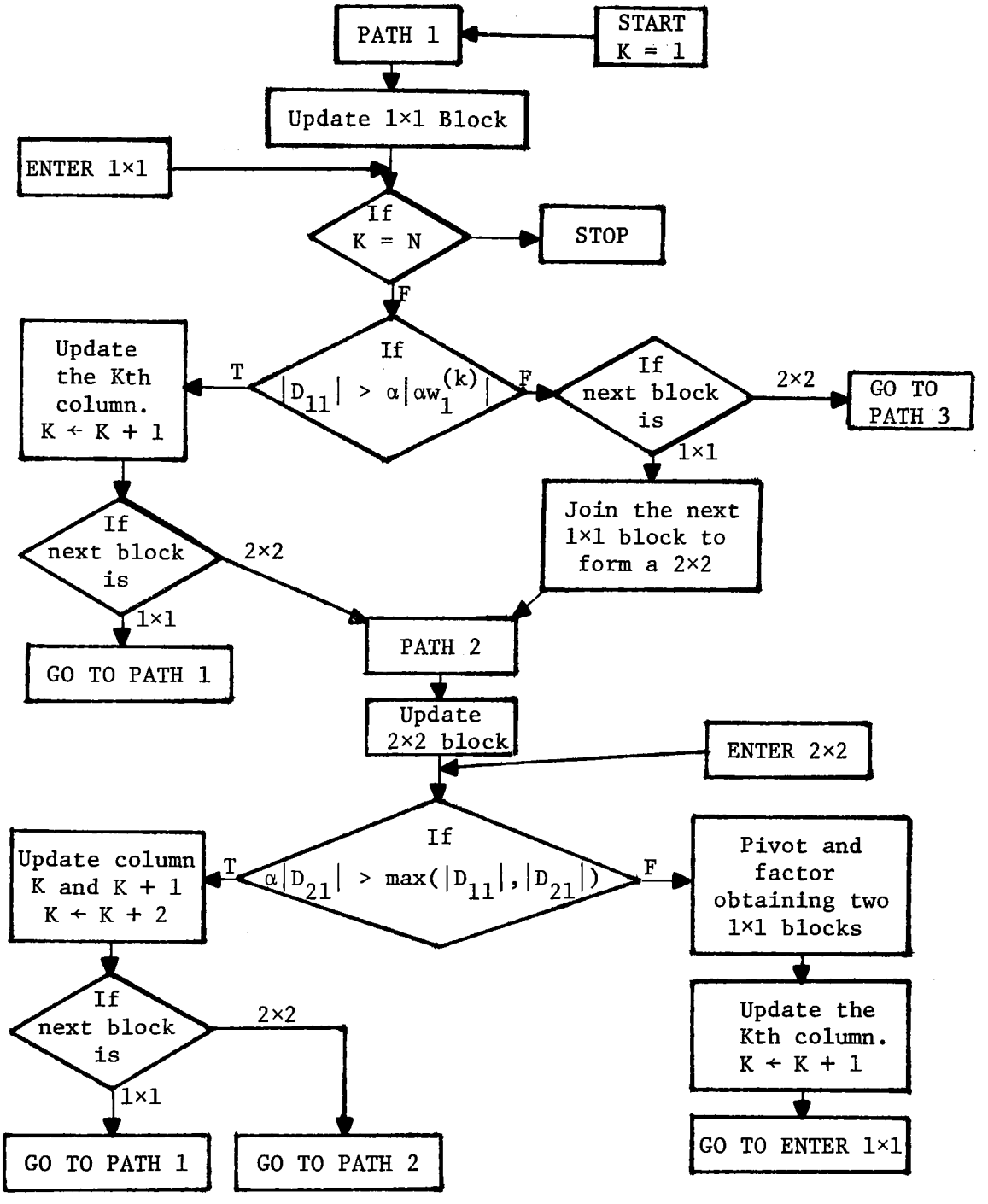
#	NITER	NFEV	$g^t g$	POSDEF	NEGCNT	FLAG
1	22	43	2×10^{-43}	T	14	0
	24	105	2×10^{-43}			0
2	21	43	2×10^{-43}	T	14	0
	21	89	8×10^{-19}			1
3	21	40	2×10^{-43}	T	13	0
	21	91	4×10^{-12}			1
4	19	45	2×10^{-43}	T	8	0
	19	81	4×10^{-15}			1
5	21	40	2×10^{-19}	T	14	0
	21	90	4×10^{-12}			1
6	19	42	2×10^{-19}	T	12	0
	21	95	2×10^{-19}			1
7	21	41	2×10^{-19}	T	14	0
	22	99	1×10^{-8}			1
8	22	41	2×10^{-43}	T	15	0
	21	101	2×10^{-43}			0
9	23	44	2×10^{-19}	T	16	0
	23	111	1×10^{-13}			1
10	19	36	2×10^{-19}	T	10	0
	21	97	4×10^{-14}			1

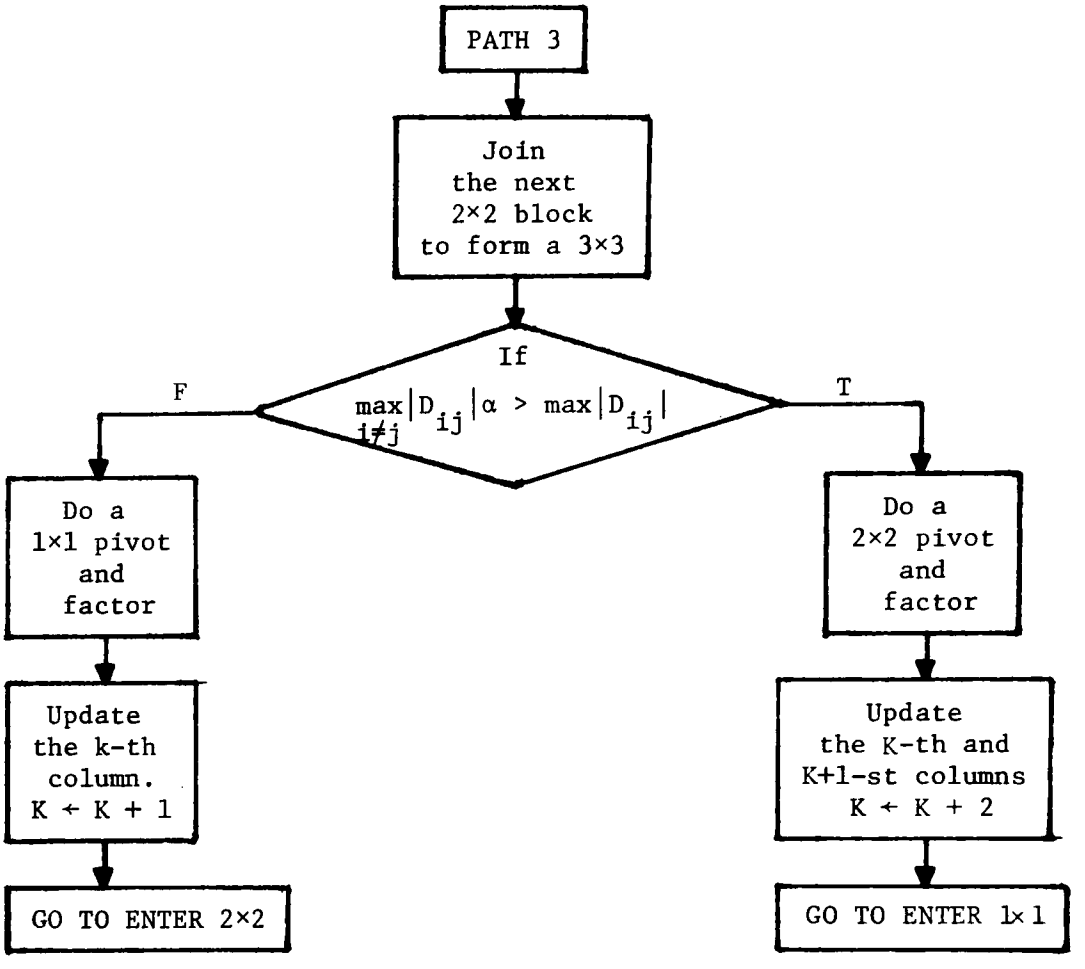
Tests using random starts were made with the other functions as well. However, with the exception of problems (3.5), (3.8), (3.9), (3.10), and (3.13), the results shown in Table 4 were consistent with results using random starts. On Beal's Function (3.10), the two algorithms behaved quite differently depending upon the starting point. The problem exhibited a lot of negative curvature. However, one algorithm would do much better than the other on one starting point, but the opposite situation would occur on another starting point.

We have compared these results with the results obtained by the algorithm of Gill and Murray [11], and have found them to be competitive. This is encouraging since the Gill and Murray algorithm has undergone a thorough development and is one of the best codes available.

The results shown here indicate that the method presented in Chapter IV is promising. Further development is needed in the practical problem areas discussed at the beginning of this section. However, the evidence so far indicates that a fully developed algorithm has the potential of being a reliable and efficient method for unconstrained optimization.

Appendix A1


START
K = 1
PATH 1
Update 1×1 Block
ENTER 1×1
If
K = N
STOP
F
If
$|D_{11}| > \alpha|\alpha w_1^{(k)}|$
T
F
Update
the Kth
column.
K ← K + 1
If
next block
is
2×2
GO TO
PATH 3
1×1
Join the next
1×1 block to
form a 2×2
If
next block
is
2×2
1×1
GO TO PATH 1
PATH 2
Update
2×2 block
ENTER 2×2
If
$\alpha|D_{21}| > \max(|D_{11}|, |D_{21}|)$
T
F
Update column
K and K + 1
K ← K + 2
Pivot and
factor
obtaining two
1×1 blocks
If
next block
is
2×2
1×1
Update the
Kth column.
K ← K + 1
GO TO PATH 1
GO TO PATH 2
GO TO ENTER 1×1


PATH 3
Join the next 2×2 block to form a 3×3
If
$\max_{i \neq j} |D_{ij}| \alpha > \max |D_{ij}|$
F
T
Do a 1×1 pivot and factor
Update the k-th column. K ← K + 1
GO TO ENTER 2×2
Do a 2×2 pivot and factor
Update the K-th and K+1-st columns K ← K + 2
GO TO ENTER 1×1

```
      SUBROUTINE SYMUPD(A,NLD,N,SIGMA,Z,CHANGE,Q,W)                          100.
      DOUBLE PRECISION SIGMA                                                 100.1
      INTEGER N,NLD                                                          100.2
      DOUBLE PRECISION A(NLD,N),Z(N),CHANGE(N),W(N)                          100.3
      INTEGER Q(N)                                                           100.4
C*************************************************************************  100.5
C                                                                            100.6
C     THIS SUBROUTINE COMPUTES THE UPDATED SYMMETRIC FACTORIZATION           100.7
C     OF AN N X N SYMMETRIC MATRIX  A  FOLLOWED BY A RANK ONE UPDATE         100.8
C     OF THE FORM  A + SIGMA*ZZ' .  IT IS ASSUMED THAT                       100.9
C                                                                            101.
C           QAQ' = MDM'                                                      101.1
C                                                                            101.2
C     WITH  D  BLOCK DIAGONAL CONSISTING OF 1 X 1 AND 2 X 2 DIAGONAL         101.3
C     BLOCKS, AND  M  OCCUPYING THE LOWER TRIANGLE OF THE PHYSICAL           101.4
C     ARRAY  A.  THE BLOCK STRUCTURE OF  D  IS INDICATED BY THE              101.5
C     ARRAY CHANGE.                                                          101.6
C       CHANGE(K) = 1 IF ENTRY K IS A 1 X 1 BLOCK                            101.7
C                 = 2 IF ENTRY K IS THE START OF A 2 X 2 BLOCK               101.8
C                 = THE DETERMINANT OF THE 2 X 2 BLOCK WHICH                 101.9
C                   STARTS AT ENTRY K-1.                                     102.
C     THE ARRAYS  A,Q,PIVOT  ARE OVERWRITTEN WITH THE UPDATED                102.1
C     FACTORIZATION                                                          102.2
C                                                                            102.3
C           MDM' = Q(A + SIGMA*ZZ')Q'                                        102.4
C                                                                            102.5
C     THE UPPER TRIANGLE OF THE PHYSICAL ARRAY  A  IS NOT ALTERED            102.6
C     IN ANY MANNER.  THUS A COPY OF THE ORIGINAL MATRIX A MAY BE            102.7
C     STORED IN THE UPPER TRIANGLE OF A  IF  A  HAS DIMENSIONS               102.8
C     N X N+1.  THE VECTOR Z IS NOT ALTERED.                                 102.9
C                                                                            103.
C                                                                            103.1
C***************                                                             103.2
C                                                                            103.3
C     A         IS A RECTANGULAR ARRAY WHOSE LEADING DIMENSION IS            103.4
C               NLD.  THIS ARRAY IS ASSUMED TO CONTAIN THE SYMMETRIC         103.5
C               MATRIX A IN FACTORED FORM AS DESCRIBED ABOVE.  THE           103.6
C               LOWER TRIANGLE OF A CONTAINS THE MATRIX M.  THE BLOCK        103.7
C               DIAGONAL MATRIX D IS STORED IN THE CORRESPONDING             103.8
C               BLOCK DIAGONAL LOCATIONS OF THE ARRAY A.  THIS IS            103.9
C               POSSIBLE SINCE IF D(I,J) (I .NE. J) IS NONZERO               104.
C               THEN M(I,J) IS ZERO.  THEREFORE,  THESE LOCATIONS            104.1
C               AS WELL AS THE DIAGONAL ENTRIES OF A MAY BE USED TO          104.2
C               TO STORE D.                                                  104.3
C                                                                            104.4
C     NLD       THE LEADING DIMENSION OF THE ARRAY A.                        104.5
C                                                                            104.6
C     N         THE DIMENSION OF THE MATRIX A.                               104.7
C                                                                            104.8
C     SIGMA     THE SCALAR DESCRIBED ABOVE.                                  104.9
C                                                                            105.
C     Z         THE N DIMENSIONAL VECTOR IN THE UPDATING FORMULA.            105.1
C                                                                            105.2
C     CHANGE THE N DIMENSIONAL ARRAY WHICH INDICATES THE BLOCK               105.3
C               STRUCTURE OF THE BLOCK DIAGONAL MATRIX D.  THE               105.4
C               CONTENTS OF THE ARRAY CHANGE ARE DESCRIBED ABOVE.            105.5
C                                                                            105.6
C     Q         AN N DIMENSIONAL INTEGER ARRAY THAT INDICATES THE            105.7
C               PIVOTING NECESSARY TO OBTAIN THE FACTORIZATION.              105.8
C                                                                            105.9
C     W         AN N DIMENSIONAL LINEAR WORK ARRAY.                          106.
C                                                                            106.1
C                                                                            106.2
C*************************************************************************  106.3
      DOUBLE PRECISION MAXNUM                                                106.4
      DOUBLE PRECISION ALFA,D11,D21,D31,D22,D32,D33,B1,B2,B3,U1,U0,          106.5
     1       T,T1,T2,DET,L1,L2                                               106.6
      INTEGER Q1,Q2,Q3,I,I0,I1,K1,K,KM1,KP1,KP2,KP3,J                        106.7
C                                                                            106.8
      ALFA=(1.0D0 + DSQRT(17.0D0))/8.0D0                                     106.9
```

```
      DO 100 J=1,N                                                          107.
        W(J)=Z(Q(J))                                                        107.1
  100 CONTINUE                                                              107.2
      K=1                                                                   107.3
  101 CONTINUE                                                              107.4
C                                                                           107.5
C     THE PROCESSING BEGINS HERE                                            107.6
C                                                                           107.7
      KP1=K+1                                                               107.8
      KP2=K+2                                                               107.9
      IF((K.LT.N).AND.(CHANGE(KP1).LT.0)) GO TO 115                         108.
C                                                                           108.1
C     THE NEXT BLOCK IS A 1 X 1 BLOCK                                       108.2
C                                                                           108.3
        T=H(K)                                                              108.4
        B1=SIGMA*T                                                          108.5
        D11=A(K,K)+B1*T                                                     108.6
        IF(KP1 .GT. N) GO TO 202                                            108.7
        DO 102 J=KP1,N                                                      108.8
          W(J)=W(J)-A(J,K)*T                                                108.9
  102   CONTINUE                                                            109.
  202   CONTINUE                                                            109.1
  103 CONTINUE                                                              109.2
C                                                                           109.3
C     ENTER 1 X 1                                                           109.4
C                                                                           109.5
          IF(K.LT.N) GO TO 104                                              109.6
            A(K,K)=D11                                                      109.7
C                                                                           109.8
C           THE DECOMPOSITION IS COMPLETE IF K=N                            109.9
C                                                                           110.
            RETURN                                                          110.1
C                                                                           110.2
  104   U1=DABS(D11)                                                        110.3
        U0=DABS(B1)                                                         110.4
        IF((U1.LT.(ALFA*U0)) .AND. (U1*DABS(SIGMA) .LE. ALFA*U0*U0))       110.5
     1              GO TO 106                                              110.6
          SIGMA=SIGMA-B1*B1/D11                                             110.7
          B1 = B1/D11                                                       110.8
          A(K,K)=D11                                                        110.9
C                                                                           111.
C     UPDATE THE K-TH COLUMN OF M.                                          111.1
C                                                                           111.2
          IF(KP1 .GT. N) GO TO 205                                          111.3
          DO 105 J=KP1,N                                                    111.4
            A(J,K)=A(J,K)+B1*W(J)                                           111.5
  105     CONTINUE                                                          111.6
  205     CONTINUE                                                          111.7
        K=KP1                                                               111.8
        GO TO 101                                                           111.9
C                                                                           112.
  106   IF((CHANGE(KP2).LE.0).AND.(KP2.LE.N)) GO TO 108                     112.1
C                                                                           112.2
C     A 2 X 2 BLOCK IS FORMED BY COMBINING THE NEXT 1 X 1 BLOCK             112.3
C     WITH BLOCK K.                                                         112.4
C                                                                           112.5
          B2=W(KP1)                                                         112.6
          T=B2                                                              112.7
          C21=B2*B1                                                         112.8
          B2=SIGMA*B2                                                       112.9
          D22=A(KP1,KP1)+B2*T                                               113.
          L1=A(KP1,K)                                                       113.1
          D22=D22+L1*D21                                                    113.2
          B2=B2+L1*B1                                                       113.3
          D21=D21+L1*D11                                                    113.4
          D22=D22+L1*D21                                                    113.5
C                                                                           113.6
C     INCLUDE INFORMATION FROM THE (K+1)-ST COLUMN OF M.                    113.7
C                                                                           113.8
          IF (KP2 .GT. N) GO TO 207                                         113.9
```

```
            DO 107 J=KP2,N                                                  114.
              W(J)=W(J)-A(J,KP1)*T                                          114.1
              A(J,K)=A(J,K)-A(J,KP1)*L1                                     114.2
  107       CONTINUE                                                        114.3
  207       CONTINUE                                                        114.4
            GO TO 117                                                       114.5
  108   CONTINUE                                                          114.6
C                                                                         114.7
C       IF THIS PORTION OF THE CODE IS REACHED WE ARE IN THE CASE OF A    114.8
C       1 X 1 SINGULAR BLOCK FOLLOWED BY A 2 X 2 BLOCK.  THIS 2 X 2       114.9
C       BLOCK IS JOINED TO THE 1 X 1 BLOCK TO FORM A 3 X 3 MATRIX D.      115.
C                                                                         115.1
            T1=W(KP1)                                                     115.2
            T2=W(KP2)                                                     115.3
            B2=SIGMA*T1                                                   115.4
            B3=SIGMA*T2                                                   115.5
            D22=A(KP1,KP1)+B2*T1                                          115.6
            D32=A(KP2,KP1)+B3*T1                                          115.7
            D33=A(KP2,KP2)+B3*T2                                          115.8
            D21=T1*B1                                                     115.9
            D31=T2*B1                                                     116.
            L1=A(KP1,K)                                                   116.1
            L2=A(KP2,K)                                                   116.2
            T=L2*D11                                                      116.3
            D33=D33+L2*(2.0D0*D31+T)                                      116.4
            D31=D31+T                                                     116.5
            D32=D32+L1*D31+L2*D21                                         116.6
            T=L1*D11                                                      116.7
            D22=D22+L1*(2.0D0*D21+T)                                      116.8
            D21=D21+T                                                     116.9
            B2=B2+L1*B1                                                   117.
            B3=B3+L2*B1                                                   117.1
            KP3=K+3                                                       117.2
C                                                                         117.3
C       INCLUDE INFORMATION FROM THE (K+1)-ST AND (K+2)-ND COLUMNS        117.4
C       OF M.                                                             117.5
C                                                                         117.6
            IF (KP3.GT.N) GO TO 209                                       117.7
            DO 109 J=KP3,N                                                117.8
              W(J)=W(J)-(A(J,KP1)*T1+A(J,KP2)*T2)                         117.9
              A(J,K)=A(J,K)-(A(J,KP1)*L1+A(J,KP2)*L2)                     118.
  109       CONTINUE                                                      118.1
  209       CONTINUE                                                      118.2
            U1=MAXNUM(D11,D22,D33,I1)                                     118.3
            U0=MAXNUM(D21,D31,D32,I0)                                     118.4
            IF (U1.LT.(ALPA*U0)) GO TO 112                                118.5
C                                                                         118.6
C       A 1 X 1 PIVOT WILL BE USED                                        118.7
C                                                                         118.8
            Q1=1                                                          118.9
            Q2=2                                                          119.
            Q3=3                                                          119.1
            CALL PIV1X1(D11,D21,D31,D22,D32,D33,B1,B2,B3,CHANGE,Q1,       119.2
     1                  Q2,Q3,I1,K,N)                                     119.3
            K1=K-1+Q1                                                     119.4
            SIGMA=SIGMA-D11*B1*B1                                         119.5
C                                                                         119.6
C       UPDATE THE K-TH COLUMN OF M                                       119.7
C                                                                         119.8
            IF(KP3.GT.N) GO TO 210                                        119.9
            DO 110 J=KP3,N                                                120.
              T=A(J,K)                                                    120.1
              A(J,K)=A(J,K1)                                              120.2
              A(J,K1)=T                                                   120.3
              A(J,K)=A(J,K)+D21*A(J,KP1)+D31*A(J,KP2)+B1*W(J)             120.4
  110       CONTINUE                                                      120.5
  210       CONTINUE                                                      120.6
            KM1=K-1                                                       120.7
```

```
C                                                                       120.8
C     INTERCHANGE THE CORRESPONDING ROWS OF M.                          120.9
C                                                                       121.
          IF (KM1 .LT. 1) GO TO 211                                     121.1
          DO 111 J=1,KM1                                                121.2
            T=A(K,J)                                                    121.3
            A(K,J)=A(K1,J)                                              121.4
            A(K1,J)=T                                                   121.5
  111     CONTINUE                                                      121.6
  211     CONTINUE                                                      121.7
          I=Q(K)                                                        121.8
          Q(K)=Q(K1)                                                    121.9
          Q(K1)=I                                                       122.
C                                                                       122.1
          A(K,K)=D11                                                    122.2
          A(KP1,K)=D21                                                  122.3
          A(KP2,K)=D31                                                  122.4
          D11=D22                                                       122.5
          D22=D33                                                       122.6
          D21=D32                                                       122.7
          B1=B2                                                         122.8
          B2=B3                                                         122.9
          K=KP1                                                         123.
          KP1=KP2                                                       123.1
          KP2=KP2+1                                                     123.2
          GO TO 117                                                     123.3
  112   CONTINUE                                                        123.4
C                                                                       123.5
C     A 2 X 2 PIVOT WILL BE USED                                        123.6
C                                                                       123.7
          Q1=1                                                          123.8
          Q2=2                                                          123.9
          Q3=3                                                          124.
          CALL PIV2X2(D11,D21,D31,D22,D32,D33,B1,B2,B3,CHANGE,SIGMA,Q1,  124.1
     1                Q2,Q3,IO,K,N)                                     124.2
          K1=K-1+Q1                                                     124.3
          K2=K-1+Q2                                                     124.4
          I=Q(K)                                                        124.5
          Q(K)=Q(K1)                                                    124.6
          Q(K1)=I                                                       124.7
          I=Q(KP1)                                                      124.8
          Q(KP1)=Q(K2)                                                  124.9
          Q(K2)=I                                                       125.
C                                                                       125.1
C     UPDATE THE K-TH AND (K+1)-ST COLUMNS OF M.                        125.2
C                                                                       125.3
          IF (KP3.GT.N) GO TO 213                                       125.4
          DO 113 J=KP3,N                                                125.5
            T=A(J,K)                                                    125.6
            A(J,K)=A(J,K1)                                              125.7
            A(J,K1)=T                                                   125.8
            T=A(J,KP1)                                                  125.9
            A(J,KP1)=A(J,K2)                                            126.
            A(J,K2)=T                                                   126.1
            A(J,K)=A(J,K)+D31*A(J,KP2)+B1*W(J)                          126.2
            A(J,KP1)=A(J,KP1)+D32*A(J,KP2)+B2*W(J)                      126.3
  113     CONTINUE                                                      126.4
  213     CONTINUE                                                      126.5
C                                                                       126.6
C     INTERCHANGE THE CORRESPONDING ROWS OF M.                          126.7
C                                                                       126.8
          KM1=K-1                                                       126.9
          DO 114 J=1,KM1                                                127.
            T=A(K,J)                                                    127.1
            A(K,J)=A(K1,J)                                              127.2
            A(K1,J)=T                                                   127.3
            T=A(KP1,J)                                                  127.4
            A(KP1,J)=A(K2,J)                                            127.5
            A(K2,J)=T                                                   127.6
  114     CONTINUE                                                      127.7
```

```
          A(K,K)=D11                                                        127.8
          A(KP1,K)=D21                                                      127.9
          A(KP1,KP1)=D22                                                    128.
          A(KP2,K)=D31                                                      128.1
          A(KP2,KP1)=D32                                                    128.2
          D11=D33                                                           128.3
          B1=B3                                                             128.4
          K=KP2                                                             128.5
          KP1=K+1                                                           128.6
          KP2=K+2                                                           128.7
          GO TO 103                                                         128.8
  115   CONTINUE                                                            128.9
C                                                                           129.
C     THE DIAGONAL BLOCK BEGINNING AT ENTRY K IS 2 X 2 .  THE UPDATED       129.1
C     DIAGONAL BLOCK D IS REQUIRED TO SATISFY                               129.2
C                                                                           129.3
C           ABS(D21)*ALFA .GT. MAX(ABS(D11) , ABS(D22)).                    129.4
C                                                                           129.5
C     IF THIS IS NOT SATISFIED THE BLOCK IS SPLIT INTO TWO 1 X 1 BLOCKS.    129.6
C                                                                           129.7
C                                                                           129.8
          T1=W(K)                                                           129.9
          T2=W(KP1)                                                         130.
          B1=SIGMA*W(K)                                                     130.1
          B2=SIGMA*W(KP1)                                                   130.2
          D11=A(K,K)+B1*W(K)                                                130.3
          D21=A(KP1,K)+B2*W(K)                                              130.4
          D22=A(KP1,KP1)+B2*W(KP1)                                          130.5
          IF (KP1.GE.N) GO TO 117                                           130.6
          DO 116 J=KP2,N                                                    130.7
            W(J)=W(J)-(A(J,K)*T1 + A(J,KP1)*T2)                             130.8
  116     CONTINUE                                                          130.9
C                                                                           131.
  117  CONTINUE                                                             131.1
C                                                                           131.2
C     ENTER 2 X 2                                                           131.3
C     THE 2 X 2 BLOCK WILL BE PROCESSED                                     131.4
C                                                                           131.5
          T1 = 0.000                                                        131.6
          U1=MAXNUM(D11,D22,T1,I1)                                          131.7
          IF (U1.GE.(ALFA*DABS(D21))) GO TO 119                             131.8
C                                                                           131.9
C     A 2 X 2 PIVOT WILL BE USED                                            132.
C                                                                           132.1
          DET=D11*D22-D21*D21                                               132.2
          CHANGE(K)=2                                                       132.3
          CHANGE(KP1)=DET                                                   132.4
          A(K,K)=D11                                                        132.5
          A(KP1,K)=D21                                                      132.6
          A(KP1,KP1)=D22                                                    132.7
          IF (KP1.EQ. N) RETURN                                             132.8
          T1=(D22*B1-D21*B2)/DET                                            132.9
          T2=(-D21*B1+D11*B2)/DET                                           133.
          IF (KP2 .GT. N) GO TO 218                                         133.1
C                                                                           133.2
C     UPDATE THE K-TH AND (K+1)-ST COLUMNS OF M.                            133.3
C                                                                           133.4
          DO 118 J=KP2,N                                                    133.5
            T = W(J)                                                        133.6
            A(J,K)=A(J,K)+T1*T                                              133.7
            A(J,KP1)=A(J,KP1)+T2*T                                          133.8
  118     CONTINUE                                                          133.9
  218     CONTINUE                                                          134.
          SIGMA=SIGMA-(T1*B1+T2*B2)                                         134.1
          K=KP2                                                             134.2
          GO TO 101                                                         134.3
  119   CONTINUE                                                            134.4
```

```
C                                                                       134.5
C     A 1 X 1 PIVOT WILL BE USED                                        134.6
C                                                                       134.7
          IF (I1.NE.2) GO TO 122                                        134.8
C                                                                       134.9
C     INTERCHANGE THE ROWS AND COLUMNS OF M IF NECESSARY.               135.
C                                                                       135.1
          T=D11                                                         135.2
          D11=D22                                                       135.3
          D22=T                                                         135.4
          T=B1                                                          135.5
          B1=B2                                                         135.6
          B2=T                                                          135.7
          I=Q(K)                                                        135.8
          Q(K)=Q(KP1)                                                   135.9
          Q(KP1)=I                                                      136.
          IF (KP2.GT.N) GO TO 220                                       136.1
          DO 120 J=KP2,N                                                136.2
            T=A(J,K)                                                    136.3
            A(J,K)=A(J,KP1)                                             136.4
            A(J,KP1)=T                                                  136.5
  120     CONTINUE                                                      136.6
  220     CONTINUE                                                      136.7
          KM1=K-1                                                       136.8
          IF (KM1 .LT. 1) GO TO 221                                     136.9
          DO 121 J=1,KM1                                                137.
            T=A(K,J)                                                    137.1
            A(K,J)=A(KP1,J)                                             137.2
            A(KP1,J)=T                                                  137.3
  121     CONTINUE                                                      137.4
  221     CONTINUE                                                      137.5
C                                                                       137.6
  122   CONTINUE                                                        137.7
C                                                                       137.8
C     PROCESS THE TWO 1 X 1 BLOCKS                                      137.9
C                                                                       138.
          CHANGE(K)=1                                                   138.1
          CHANGE(KP1)=1                                                 138.2
          D22=D22-(D21*D21)/D11                                         138.3
          D21=D21/D11                                                   138.4
          B2=B2-B1*D21                                                  138.5
          B1=B1/D11                                                     138.6
          IF (KP2.GT.N) GO TO 223                                       138.7
C                                                                       138.8
C     UPDATE THE K-TH COLUMN OF M.                                      138.9
C                                                                       139.
          DO 123 J=KP2,N                                                139.1
            A(J,K)=A(J,K)+D21*A(J,KP1)+B1*W(J)                          139.2
  123     CONTINUE                                                      139.3
  223     CONTINUE                                                      139.4
          A(K,K)=D11                                                    139.5
          A(KP1,K)=D21                                                  139.6
          SIGMA=SIGMA-B1*D11*B1                                         139.7
          D11=D22                                                       139.8
          B1=B2                                                         139.9
          K=KP1                                                         140.
          KP1=KP2                                                       140.1
          KP2=K+2                                                       140.2
          GO TO 103                                                     140.3
      END                                                               140.4
```

```
      SUBROUTINE PIV1X1(D11,D21,D31,D22,D32,D33,B1,B2,B3,CHANGE,
     1                  Q1,Q2,Q3,I1,K,N)
      DOUBLE PRECISION D11,D21,D31,D22,D32,D33,B1,B2,B3
      INTEGER Q1,Q2,Q3,I1,K,N
      DOUBLE PRECISION CHANGE(N)
C*****************************************************************
C
C
C     THIS SUBROUTINE PERFORMS A 1 X 1 PIVOT.  GIVEN A 3 X 3
C     SYMMETRIC MATRIX D=(DIJ)  WHICH SATISFIES THE 1 X 1 PIVOT
C     CRITERIA WITH D(I1,I1) AS THE PIVOT ELEMENT.  THE 3 X 3 MATRIX  D
C     IS PERMUTED TO BRING D(I1,I1)  TO THE (1,1)  POSITION  AND
C     THEN THE FIRST STEP OF THE FACTORIZATION IS DONE IN PLACE.
C
C
C*****************************************************************
C
      DOUBLE PRECISION T
      INTEGER KP1,KP2
      KP1=K+1
      KP2=K+2
      GO TO (10,20,30),I1
C
C     THE MAX ELEMENT IS D22
C
   20   T=D11
        D11=D22
        D22=T
C
C
        T=D32
        D32=D31
        D31=T
C
        T=B2
        B2=B1
        B1=T
        Q1=2
        Q2=1
      GO TO 10
C
C     THE MAX ELEMENT IS D33
C
   30   T=D11
        D11=D33
        D33=T
C
        T=D21
        D21=D32
        D32=T
C
        T=B1
        B1=B3
        B3=T
C
        Q1=3
        Q3=1
C
C     THE MAX ELEMENT IS D11
C
   10 D22=D22-(D21*D21)/D11
      D32=D32-(D31*D21)/D11
      D33=D33-(D31*D31)/D11
      B1=B1/D11
      B2=B2-B1*D21
      B3=B3-B1*D31
      D21=D21/D11
      D31=D31/D11
      CHANGE(K)=1.
      RETURN
      END
```

```
      SUBROUTINE PIV2X2(D11,D21,D31,D22,D32,D33,B1,B2,B3,CHANGE,SIGMA,    147.6
     1                  Q1,Q2,Q3,IO,K,N)                                  147.7
      DOUBLE PRECISION D11,D21,D22,D31,D32,D33,B1,B2,B3,SIGMA            147.8
      INTEGER Q1,Q2,Q3,IO,K,N                                            147.9
      DOUBLE PRECISION CHANGE(N)                                         148.
C*********************************************************************  148.1
C                                                                        148.2
C                                                                        148.3
C     THIS SUBROUTINE PERFORMS A 2 X 2 PIVOT ON THE 3 X 3 MATRIX         148.4
C     D = (DIJ).  THE MAXIMUM OFF-DIAGONAL ELEMENT IS BROUGHT TO THE     148.5
C     (2,1)  POSITION.  ITS ORIGINAL LOCATION IS INDICATED BY THE        148.6
C     VARIABLE IO:                                                       148.7
C                                                                        148.8
C           IO=1  D21 IS THE MAX ELEMENT                                 148.9
C           IO=2  D31 IS THE MAX ELEMENT                                 149.
C           IO=3  D32 IS THE MAX ELEMENT                                 149.1
C                                                                        149.2
C     THE FIRST STEP OF THE FACTORIZATION OF THE MATRIX (DIJ) IS         149.3
C     CARRIED OUT IN PLACE USING THE 2X2 PIVOT.                          149.4
C                                                                        149.5
C*********************************************************************  149.6
C                                                                        149.7
      INTEGER KP1,KP2                                                    149.8
      DOUBLE PRECISION S,T,DET                                           149.9
C                                                                        150.
      KP1=K+1                                                            150.1
      KP2=K+2                                                            150.2
      GO TO (10,20,30),IO                                                150.3
   20 CONTINUE                                                           150.4
C                                                                        150.5
C     D31 IS THE MAX ELEMENT                                             150.6
C                                                                        150.7
        T=D22                                                            150.8
        D22=D33                                                          150.9
        D33=T                                                            151.
C                                                                        151.1
        T=D21                                                            151.2
        D21=D31                                                          151.3
        D31=T                                                            151.4
C                                                                        151.5
        T=B2                                                             151.6
        B2=B3                                                            151.7
        B3=T                                                             151.8
C                                                                        151.9
        Q2=3                                                             152.
        Q3=2                                                             152.1
      GO TO 10                                                           152.2
   30 CONTINUE                                                           152.3
C                                                                        152.4
C     D32 IS THE MAX ELEMENT                                             152.5
C                                                                        152.6
        T=D11                                                            152.7
        D11=D22                                                          152.8
        D22=D33                                                          152.9
        D33=T                                                            153.
C                                                                        153.1
        T=D21                                                            153.2
        D21=D32                                                          153.3
        D32=D31                                                          153.4
        D31=T                                                            153.5
C                                                                        153.6
        T=B1                                                             153.7
        B1=B2                                                            153.8
        B2=B3                                                            153.9
        B3=T                                                             154.
C                                                                        154.1
        Q1=2                                                             154.2
        Q2=3                                                             154.3
        Q3=1                                                             154.4
   10 CONTINUE                                                           154.5
```

```
C                                                                       154.6
C     D21 IS THE MAX ELEMENT                                            154.7
C     THE 2 X 2 PIVOT IS DONE HERE                                      154.8
C                                                                       154.9
      DET=D11*D22-D21*D21                                               155.
      T=(D22*D31-D21*D32)/DET                                           155.1
      S=(-D21*D31+D11*D32)/DET                                          155.2
      B3=B3-(T*B1+S*B2)                                                 155.3
      D33=D33-(T*D31+S*D32)                                             155.4
      D31=T                                                             155.5
      D32=S                                                             155.6
      T=(D22*P1-D21*B2)/DET                                             155.7
      S=(-D21*B1+D11*B2)/DET                                            155.8
      SIGMA=SIGMA-(T*B1+S*B2)                                           155.9
      B1=T                                                              156.
      B2=S                                                              156.1
      CHANGE(K)=2                                                       156.2
      CHANGE(KP1)=DET                                                   156.3
      CHANGE(KP2) = 1                                                   156.4
      RETURN                                                            156.5
      END                                                               156.6
```

```
      DOUBLE PRECISION FUNCTION MAXNUM(A,B,C,I)                         156.7
      DOUBLE PRECISION A,B,C                                            156.8
      INTEGER I                                                         156.9
C*********************************************************************  157.
C                                                                       157.1
C     THIS FUNCTION FINDS THE MAXIMUM OF THE ABSOLUTE VALUES OF A,B,C   157.2
C     AND INDICATES WHICH OF THE VALUES IS SELECTED BY SETTING          157.3
C     I = 1,2,3  RESPECTIVELY.                                          157.4
C                                                                       157.5
C*********************************************************************  157.6
      DOUBLE PRECISION S,T                                              157.7
        I=1                                                             157.8
        T=DABS(A)                                                       157.9
        S=DABS(B)                                                       158.
        IF (S.LE.T) GO TO 10                                            158.1
          T=S                                                           158.2
          I=2                                                           158.3
   10   CONTINUE                                                        158.4
        S=DABS(C)                                                       158.5
        IF (S.LE.T) GO TO 20                                            158.6
          T=S                                                           158.7
          I=3                                                           158.8
   20   CONTINUE                                                        158.9
        MAXNUM=T                                                        159.
      RETURN                                                            159.1
      END                                                               159.2
```

```
      SUBROUTINE SOLVE(A,NLD,N,CHANGE,Q,X,IFAIL)                        159.3
      INTEGER N,NLD,IFAIL                                               159.4
      DOUBLE PRECISION A(NLD,N),X(N),CHANGE(N)                          159.5
      INTEGER Q(N)                                                      159.6
C*****************************************************************     159.7
C                                                                       159.8
C     THIS SUBROUTINE COMPUTES THE SOLUTION TO AX = B.                  159.9
C                                                                       160.
C     THE MATRIX A IS ASSUMED TO BE IN THE FACTORED FORM                160.1
C                                                                       160.2
C           QAQ' = MDM'                                                 160.3
C                                                                       160.4
C     WHERE M IS BLOCK UNIT LOWER TRIANGULAR AND D IS BLOCK             160.5
C     DIAGONAL WITH 1X1 AND 2X2 DIAGONAL BLOCKS.  IT IS                 160.6
C     ASSUMED THAT M, D ARE OUTPUT FROM THE ROUTINE SYMUPD              160.7
C     AND THAT THAT THESE ARRAYS ARE STORED IN THE LOWER                160.8
C     TRIANGLE OF A.  ON INPUT THE ARRAY  X  CONTAINS THE               160.9
C     RIGHT HAND SIDE  B  AND ON OUTPUT  X  CONTAINS THE                161.
C     SOLUTION VECTOR.  IFAIL  =  1  IF THE SYSTEM IS                   161.1
C     SINGULAR (NO SOLUTION IN THIS CASE)  OTHERWISE  IFAIL             161.2
C     IS RETURNED WITH THE VALUE  0  (A SOLUTION WAS OBTAINED).         161.3
C                                                                       161.4
C                                                                       161.5
C***************                                                        161.6
C                                                                       161.7
C                                                                       161.8
C     A         THE ARRAY A IS RECTANGULAR WITH LEADING DIMENSION NLD.  161.9
C               THE SECOND DIMENSION MUST BE GREATER THAN OR EQUAL TO N.  162.
C               THE ARRAY A IS ASSUMED TO HAVE THE FACTORIZATION OF THE  162.1
C               MATRIX A AS DESCRIBED IN THE SUBROUTINE SYMUPD.          162.2
C                                                                       162.3
C     NLD       THE LEADING DIMENSION OF THE ARRAY A.                   162.4
C                                                                       162.5
C     N         THE DIMENSION OF THE MATRIX A.                          162.6
C                                                                       162.7
C     CHANGE    AN N DIMENSIONAL VECTOR WHICH CONTAINS A DESCRIPTION    162.8
C               OF THE BLOCK STRUCTURE OF D,  AND THE DETERMINANT       162.9
C               OF EACH 2X2 DIAGONAL OF D.  SEE THE DOCUMENTATION FOR   163.
C               THE SUBROUTINE SYMUPD FOR A MORE COMPLETE DESCRIPTION   163.1
C               OF THE CONTENTS OF CHANGE.                              163.2
C                                                                       163.3
C     Q         AN N DIMENSIONAL INTEGER ARRAY WHICH CONTAINS THE       163.4
C               PIVOTING USED TO OBTAIN THE FACTORIZATION OF A.         163.5
C                                                                       163.6
C     X         AN N DIMENSIONAL VECTOR.  THE CONTENTS OF X ARE         163.7
C               DESCRIBED ABOVE.                                        163.8
C                                                                       163.9
C     IFAIL     AN INTEGER VARIABLE THAT INDICATES WHEN A IS SINGULAR.  164.
C               THE CONTENTS OF IFAIL ARE DESCRIBED ABOVE.              164.1
C                                                                       164.2
C                                                                       164.3
C*****************************************************************     164.4
C                                                                       164.5
      DOUBLE PRECISION T,S                                              164.6
      INTEGER I,J,K,IP1,IP2                                             164.7
      DOUBLE PRECISION W(50)                                            164.8
      IFAIL = 0                                                         164.9
      DO 10 J = 1,N                                                     165.
         W(J) = X(Q(J))                                                 165.1
   10 CONTINUE                                                          165.2
C                                                                       165.3
C     BACKSOLVE THE LOWER TRIANGULAR SYSTEM AND INVERT THE DIAGONAL     165.4
C     BLOCKS.                                                           165.5
C                                                                       165.6
      I = 1                                                             165.7
   20 IF (I .GE. N) GO TO 60                                            165.8
      IP1 = I + 1                                                       165.9
      IF (CHANGE(IP1) .GT. 0) GO TO 40                                  166.
      IF (CHANGE(IP1) .EQ. 000) GO TO 1000                              166.1
```

```
C                                                                       166.2
C     WE HAVE A 2 X 2 PIVOT AT STEP I                                   166.3
C                                                                       166.4
      IP2 = I + 2                                                       166.5
      S = W(I)                                                          166.6
      T = W(IP1)                                                        166.7
      IF (IP2 .GT. N) GO TO 130                                         166.8
      DO 30 J = IP2,N                                                   166.9
          W(J) = W(J) - (S*A(J,I) + T*A(J,IP1))                         167.
   30 CONTINUE                                                          167.1
  130 CONTINUE                                                          167.2
      W(I)   = (A(IP1,IP1)*S - A(IP1,I)*T)/CHANGE(IP1)                  167.3
      W(IP1) = (-A(IP1,I)*S + A(I,I)*T)/CHANGE(IP1)                     167.4
      I = IP2                                                           167.5
      GO TO 20                                                          167.6
C                                                                       167.7
   40 CONTINUE                                                          167.8
C                                                                       167.9
C     WE HAVE A 1 X 1 PIVOT AT STEP I                                   168.
C                                                                       168.1
      T = W(I)                                                          168.2
      DO 50 J = IP1,N                                                   168.3
          W(J) = W(J) - A(J,I)*T                                        168.4
   50 CONTINUE                                                          168.5
      IF (A(I,I) .EQ. 0.0D0) GO TO 1000                                 168.6
      W(I) = W(I)/A(I,I)                                                168.7
      I = IP1                                                           168.8
      GO TO 20                                                          168.9
C                                                                       169.
   60 I = N                                                             169.1
C                                                                       169.2
C     INVERT THE LAST DIAGONAL BLOCK AND INITIALIZE                     169.3
C     FOR THE FORWARD SOLUTION                                          169.4
C                                                                       169.5
      IF (CHANGE(I) .GT. 0.0D0) GO TO 65                                169.6
      IF (CHANGE(I) .EQ. 0D0) GO TO 1000                                169.7
C                                                                       169.8
C    THE LAST BLOCK IS 2 X 2                                            169.9
C     IT HAS ALREADY BEEN INVERTED                                      170.
C                                                                       170.1
          IP1 = I - 1                                                   170.2
          I   = I - 2                                                   170.3
          GO TO 70                                                      170.4
C                                                                       170.5
   65 CONTINUE                                                          170.6
C                                                                       170.7
C     THE LAST BLOCK IS 1 X 1                                           170.8
C                                                                       170.9
          IF (A(N,N) .EQ. 0.0D0) GO TO 1000                             171.
          W(N) = W(N)/A(N,N)                                            171.1
          IP1  = I                                                      171.2
          I    = I - 1                                                  171.3
C                                                                       171.4
   70 CONTINUE                                                          171.5
C                                                                       171.6
C     FORWARD SOLVE THE REMAINING UPPER TRIANGULAR SYSTEM               171.7
C                                                                       171.8
      IF (I .LE. 0) GO TO 1001                                          171.9
      IF (CHANGE(I) .GT. 0.0D0) GO TO 90                                172.
C                                                                       172.1
C     2 X 2 PIVOT                                                       172.2
C                                                                       172.3
      IP2 = IP1                                                         172.4
      IP1 = I                                                           172.5
      I   = I - 1                                                       172.6
      DO 80 J = IP2,N                                                   172.7
          W(IP1) = W(IP1) - A(J,IP1)*W(J)                               172.8
          W(I)   = W(I) - A(J,I)*W(J)                                   172.9
   80 CONTINUE                                                          173.
```

```
      IP1 = I                                                          173.1
      I   = I - 1                                                      173.2
      GO TO 70                                                         173.3
C                                                                      173.4
C     1 X 1  PIVOT                                                     173.5
C                                                                      173.6
   90 DO 100 J = IP1,N                                                 173.7
         W(I) = W(I) - A(J,I)*W(J)                                     173.8
  100 CONTINUE                                                         173.9
      IP1 = I                                                          174.
      I   = I -1                                                       174.1
      GO TO 70                                                         174.2
C                                                                      174.3
C                                                                      174.4
 1000 CONTINUE                                                         174.5
C                                                                      174.6
C     THE MATRIX IS SINGULAR                                           174.7
C                                                                      174.8
      IFAIL = 1                                                        174.9
      RETURN                                                           175.
C                                                                      175.1
 1001 CONTINUE                                                         175.2
C                                                                      175.3
C     THIS IS THE NORMAL RETURN . . .  A SOLUTION WAS FOUND            175.4
C                                                                      175.5
      DO 110 J = 1,N                                                   175.6
         X(Q(J)) = W(J)                                                175.7
  110 CONTINUE                                                         175.8
      RETURN                                                           175.9
      END                                                              176.
```

Acknowledgments

First of all, I wish to express my deep appreciation to Professor James R. Bunch. He has generously given his time, energy, and valuable advice to help me with this dissertation. He has also given me many fine opportunities that have enriched both my personal and academic lives. Working under his guidance has been a very rewarding experience.

I would also like to thank the other members of the doctoral committee: Professors John W. Evans, John A. Trangenstein, Walter A. Burkhard, and Michael L. Fredman. A special note of thanks goes to Dr. Jorge J. Moré for stimulating my interest in optimization and for many discussions and helpful suggestions.

There are many people who have not been directly involved in the preparation of this thesis, and yet who have influenced my thoughts. They have been my teachers along the way, and my gratitude for their gift of knowledge is endless. A few people deserve special mention. Professor John W. Evans and Professor Murray Rosenblatt at University of California, San Diego gave me the opportunity to gain computing experience by working on their research projects. Dr. Gary Leaf and Dr. Mike Minkoff provided a stimulating summer project at Argonne National Laboratory in 1975.

In addition, I would like to express my appreciation to the National Science Foundation for financial support under NSF grants MCS 76-03139 and MCS 76-17548. I also appreciate the financial support from the Applied Mathematics Division at Argonne National Laboratory.

Finally, I want to thank Ms. Judy Beumer for an excellent job of typing and for her infinite patience.

References

[1] Aasen, J. O., "On the Reduction of a Symmetric Matrix to Tridiagonal Form," BIT, 11, (1971), 233-242.

[2] Armijo, L., "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives," Pac. J. Math. 16, (1966), 1-3.

[3] Bunch, J. R., "Analysis of the Diagonal Pivoting Method," Siam J. Numer. Anal., 8, (1971), 656-680.

[4] Bunch, J. R. and Kaufman, L., "Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems," Math. of Comp., 31, (1977), 162-179.

[5] Bunch, J. R. and Parlett, B. N., "Direct Methods for Solving Symmetric Indefinite Systems of Linear Equations," Siam J. Numer. Anal., 8, (1971), 639-655.

[6] Dahlquist, G. and Björk, Å., Numerical Methods, Prentice-Hall Inc., New Jersey, 1974.

[7] Dennis, J. E. and Moré, J., "Quasi Newton Methods, Motivation and Theory," SIAM Review, 19, (1977), 46-89.

[8] Fletcher, R. and Freeman, T. L., "A Modified Newton Method for Minimisation," U. of Dundee Rept. No. 7, (1975).

[9] Fletcher, R. and Powell, M. J. D., "Modifications of LDL^t Factorizations," Math. of Comp., 28, (1974), 1067-1087.

[10] Forsythe, G. E. and Moler, B. C., Computer Solutions of Linear Algebraic Systems, Prentice-Hall, 1967.

[11] Gill, P. E. and Murray, W., "Newton Type Methods for Unconstrained and Linearly Constrained Optimization," Math. Prog., 7, (1974), 311-350.

[12] Gill, P. E. and Murray, W., "Safeguarded Steplength Algorithms for Optimization Using Descent Methods," NPL Rept. NAC37, (1974).

[13] Gill, P. E., Golub, G. H., Murray, W., Saunders, M. A., "Methods for Modifying Matrix Factorizations," Math. of Comp., 28, (1974), 505-535.

[14] Gill, P. E., Murray, W., Saunders, M. A., "Methods for Computing and Modifying the LDV Factors of a Matrix," Math. of Comp., 29, (1975), 1051-1077.

[15] Hillstrom, K. E., "A Simulation Test Approach to the Evaluation and Comparison of Unconstrained Nonlinear Optimization Algorithms," Rep. ANL-76-20, Argonne National Laboratory, Argonne, IL, (1976).

[16] McCormick, G., "Second Order Convergence Using a Modified Armijo Step Size Rule for Function Minimization," The George Washington University Serial T-328, (1976).

[17] Ortega, J. M. and Rheinbolt, W. C., Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, (1970).

[18] Powell, M. J. D., "A New Algorithm for Unconstrained Optimization," Nonlinear Programming, J. B. Rosen, O. L. Mangasarian and K. Ritter, eds., Academic Press, New York, (1970).

[19] Stewart, G. W., Introduction to Matrix Computations, Academic Press, New York, 1973.

[20] Wilkinson, J. H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

Distribution for ANL-77-49

Internal:

B. Ancker-Johnson
R. J. Royston
P. Witkowski (10)
C. E. Till
D. C. Sorensen (74)
A. B. Krisciunas
ANL Contract Copy
ANL Libraries (5)
ANL TIS Files (6)

External:

ERDA-TIC, for distribution per UC-32 (201)
Manager, Chicago Operations Office
Chief, Chicago Patent Group
President, Argonne Universities Association
Applied Mathematics Division Review Committee:
    P. J. Eberlein, SUNY at Buffalo
    G. Estrin, U. California, Los Angeles
    W. M. Gentleman, U. Waterloo
    J. M. Ortega, NASA Langley Research Center
    E. N. Pinson, Bell Telephone Labs.
    S. Rosen, Purdue U.